筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

MeeGo をベースにした次世代組み込みデバイス向けユーザ体験開発 ―データ共有システムのサーバーソフトウェアの設計開発―

刘宏超

(コンピュータサイエンス専攻)

指導教員　田中二郎

２０１２年　３月

概要

本報告書においては，MeeGo（Intel 社を中心に開発された Moblin と Nokia 社の Maemo プロジェクトを統合した次世代コンピューター機器のためのオープンソースソフトウェアプラットフォームである）向けのシステムの企画を立案し，その一つであるデータ共有システムのサーバーソフトウェアの設計開発を行い，評価を実施した．

企画では MeeGo 向けソフトウェアの開発を前提として，さまざまな視点から六つの計画を提案した．MeeGo の多種多様な機器に対応している特徴を活かすため，案の中から「データ共有システム」を取り上げた．

技術調査では，著者がクライアントソフトウェアの実現性，クライアントソフトウェアとサーバーソフトウェアとの通信方式と暗号化方法について，技術的な制約を調査し，対策を検討し実現方式の設計を行った．その後，データ取得用デモプログラムを作成した．また，選定した計画案の実現性を確認した上で，本システムの通信方式はフレームワークを利用せず独自の方法でソケット通信を実装し， XML データの解析・処理を行うように設計した．それに SSL の設計に基づいて本システムに適した暗号化方法を設計した．

開発では，著者がサーバーソフトウェアとクライアントデモプログラムの開発を行い，非機能要件である信頼性，効率性やセキュリティなどを考慮した設計や実装に工夫を行った．

評価では，作成したサーバーソフトウェアの信頼性について実験を行い検証した．実験結果より，多くのクライアントは同時に接続しても正しく処理できること，処理時間は予定通りであることが確認できた．

本報告書では，これらの研究開発の成果の概要について報告するとともに，特に著者が担当した部分について報告するものである．

目次

図目次

表目次

第1章　はじめに

筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻高度IT人材育成のための実践的ソフトウェア開発専修プログラム（以下，高度ITコース）における特定課題研究で，インテル株式会社（以下，インテル社）を中心に開発された次世代共通プラットフォームであるMeeGo[1]（以下，MeeGo）を対象として，著者と筑波大学システム情報工学研究科コンピュータサイエンス専攻の許先華，森本悠矢，刘建宇の学生チーム（以下，チーム）はソリューション企画の立案，設計開発を行った．

本報告書はMeeGo向けソリューションの計画立案とデータ共有システムの設計開発について，特に著者が設計，開発したサーバーソフトウェアについて報告するものである．

本章は MeeGo の説明を始めプロジェクトの目的と報告書の構成について述べる．

1.1　MeeGo について(1)

MeeGo は, 次世代コンピューター機器のためのオープンソースソフトウェアプラットフォームとして，インテル社を中心に開発された Moblin と Nokia Corporation の Maemo プロジェクトを統合したもので，Linux Foundation の下，Intel のサポートを受けて開発されている．以下に MeeGo の特徴を示す．

- パフォーマンスの最適化や高度な機能が提供されているため，複雑な計算やグラフィックを必要とするアプリケーションやオンラインサービスの開発が可能である．
- Linux として小型機器やモバイル機器に最適化されているため，幅広い Linux アプリケーションが利用できる．
- Linux Foundation が管理しているオープンソースプロジェクトである．
- 一つのソースコードはスマートフォン，ノートブック，車載機器などさまざまなデバイスにも流用可能である．
- 主に携帯端末のオペレーティングシステムとして使用する. インテル社の ATOM[2]系や ARM[3]系で動作できる．

1.2　プロジェクトの目的

本プロジェクトでは，インテル社からの技術支援を受けながら，学生チームでソリューションの計画立案，設計，開発や評価を行う．インテル社のビジョンの一つであるコンティニュアム・コンピューティングを実現し，パソコン，タブレットやスマートフォンなどの端末向け，場所が変わっても，同じ利用感あるいは利用形態を継続できるソフトウェアの開発を目標とした．

1.3　報告書の構成

本報告書の構成は以下に示す．

第２章では，プロジェクトの構成，役割分担，計画立案の経緯と立案した計画案の選定について述べる．

---

(1)　この部分については MeeGo の HP(http://www.meego.jp/)から引用している．

第３章では，データ共有システムの要件，構成や開発時の検討事項などのシステム概要について述べる．

第４章では，システム詳細モジュールの構成，また著者が担当したモジュールについて述べる．

第５章では，著者が担当した部分であるサーバーソフトウェアの詳細について述べる．

第６章では，第５章以外に，著者が担当した部分について述べる．

第７章では，著者が担当した部分であるサーバーソフトウェアの評価について述べる．

第８章では，著者が担当したプログラムの開発計画と実績，技術調査，開発と評価フェーズの振り返りについて述べる．

第９章では，本報告書の結論を述べる．

第2章　　プロジェクトの概要

本プロジェクトは MeeGo 向けソリューションソフトウェアの計画立案から技術調査，立案選定，システムの設計から開発，評価まで行った．各個開発フェーズの作業内容は図 2-1 に示す．

図 2-1　　開発フェーズと作業内容

また，プロジェクト体制を図 2-2 に示す．

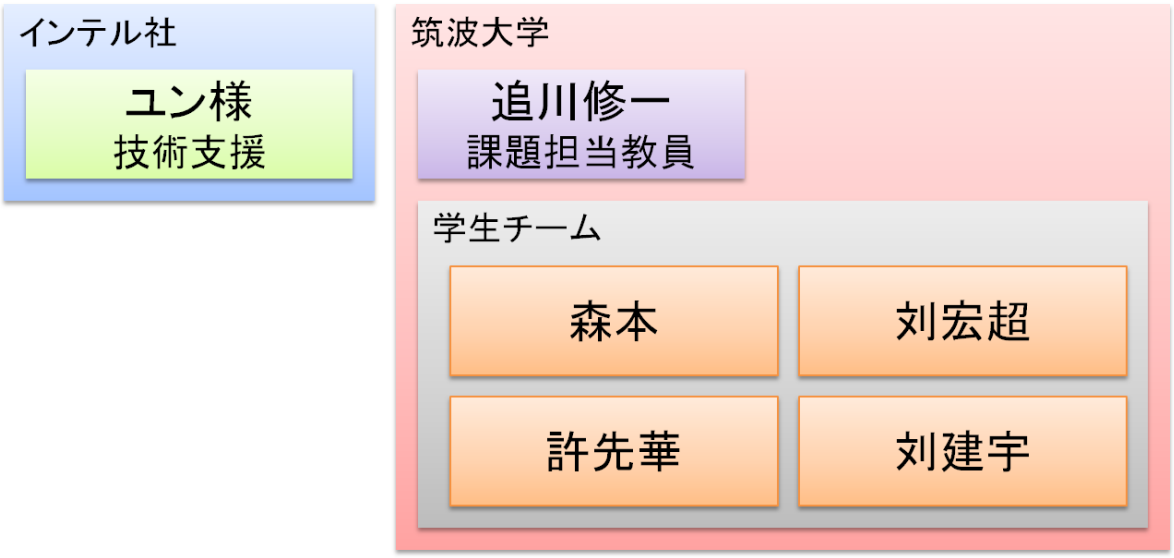


図 2-2　　プロジェクト体制

インテル社のユン様は技術的な支援と成果物のレビュー・確認を担当し，また筑波大学側は，課題担当教員の追川先生と学生４名のチームである．

2.1　役割の分担

プロジェクト運営と開発フェーズの役割分担を図 2-3 に示す．

図 2-3　プロジェクト運営と開発フェーズの役割分担

著者が開発フェーズでサーバーソフトウェアの内部処理と通信処理モジュールの開発を担当していた．また，割り振れた役割を担当する以外，チーム内の技術担当として，テクニック問題の解決，技術的な先行調査とデモプログラム作成など含めて，他のチームメンバーをサポートしていた．

著者が担当した部分の詳細説明は第５，６章にて述べる．

2.2　計画立案

計画立案フェーズでは，六つのアプリケーションとシステムの計画を立案した．以下は提案した計画の内容，特に著者が提案した計画の説明，計画案の選定基準と立案の選定経緯について述べる．

2.2.1 立案した計画案

立案した計画案は以下の六つである.

1. 動画配信アプリケーション
   MeeGo 端末の I/O デバイスを最大限活用し, ライブ生放送機能と端末間動画を切換えて視聴できる動画配信機能を持つアプリケーション.
2. 健康管理アプリケーション
   食事や運動量など健康関連データを収集する，またそれらのデータを利用し，利用者の健康管理をサポートするアプリケーション.
3. MeeGo 既存 QML(2)コンポーネントの改造・拡張
   QML 言語で開発した MeeGo 既存の UI ライブラリを改造・拡張し，より多く，より便利な UI ライブラリの提供.
4. ビジュアルプログラミングシステム
   タブレットや携帯などデバイス向け開発環境を開発することで，テキストより直感的にプログラムできるシステム.
5. WEB ミーティングシステム
   MeeGo 端末向け最適化した WEB ミーティングシステム.
6. データ共有システム（著者の提案）
   MeeGo デバイス間のデータを共有するシステム.
   例えば，アドレス帳，ブラウザの閲覧履歴，メモやカレンダーなどの情報をサーバーへ自動保存し，共有できるようにする．利用者はネットワークを経由して，手動でデータを転送すること無く，いつも自分が所有するデバイスの最新情報を取得できる.

2.2.2 計画案の選定

MeeGo 上で動作するソフトウェアの開発を第一前提として，MeeGo 端末のハードウェア制限やチームメンバーの技術能力など総合的に検討した．計画案の選定基準を以下に示す.

- MeeGo 端末のハードウェア制限を配慮する
- チームメンバーの開発能力と開発時間を配慮する
- MeeGo に対してよりユニークな提案をする

立案の選定する段階では，各個計画案に対して，以上の選定基準より評価を行った．評価結果を表 2-1 に示す.

(2) QML とはユーザーインタフェースの記述に特化したスクリプト言語である.

表 2.1　計画案の評価

計画案	制約条件	総合評価
動画配信アプリケーション	サーバーが必要 著作権問題	単なる動画配信で， 開発価値が薄い
健康管理アプリケーション	サーバーが必要 デバイスに依存 医学知識が必要	同様機能の物は多いため， 開発価値が薄い
QML の改造・拡張	QML 知識が必要 デモの作成は必要	メンバーの知識が不足， 時間がかかる
WEB ミーテイング	サーバーが必要	似たようなものが多い 開発価値が薄い
ビジュアルプログラミング	無し	目標不明 利用者が少ない 評価が難しい
データ共有システム	サーバーが必要	利用者の手間を減らすため， 想定利用者が多い

これらの中で，データ共有システムは，本プロジェクトの目的であるコンティニュアム・コンピューティングを実現することと最も適している．また，開発規模，予想開発期間も適しており，さらに想定する利用者も多いため，最終的に決定した．

第3章　　データ共有システムの概要

本章では，開発したデータ共有システムの概要について述べる．

3.1　　システムの目的

人々が普段の生活の中で使っているデバイス（パソコンやテレビ，携帯電話，車載端末など）は大量のデータを扱っている．デバイス間のデータ共有はまだ不便であり，共有しにくい面も存在する．本システムはデバイス間でデータの自動共有を行い，このような不便を解決するためのものである．

システムの概要を図 3-1 に示す．

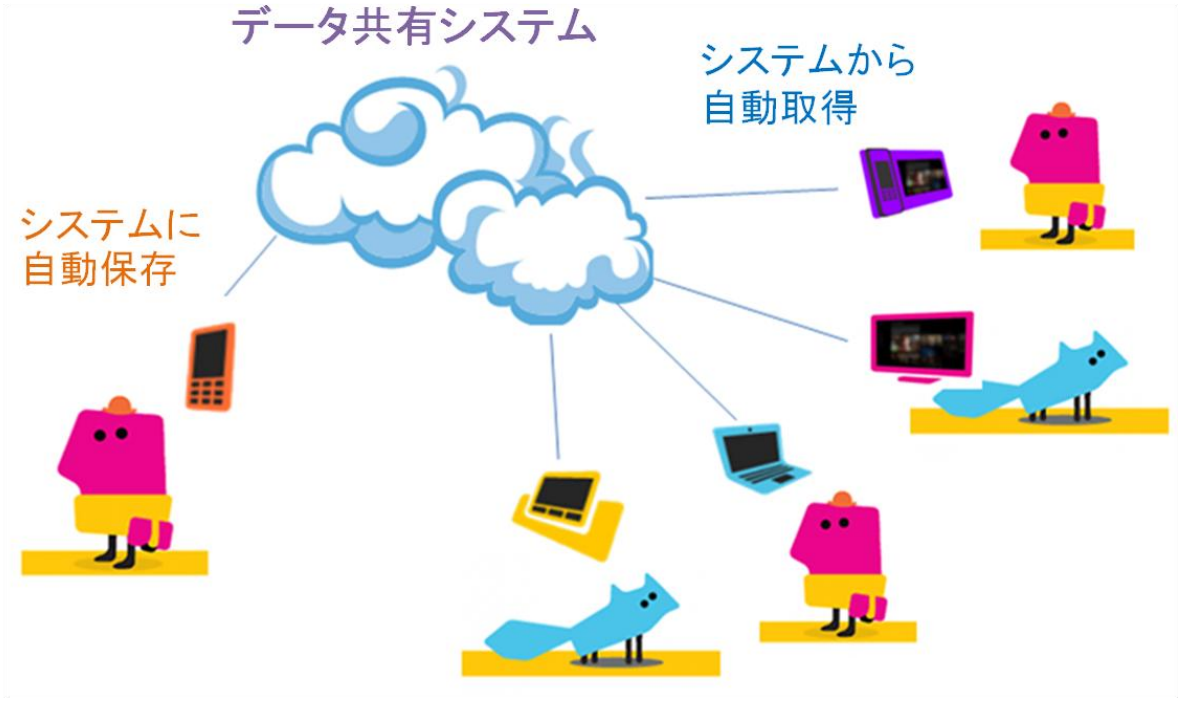


図 3-1　　システム概要図

デバイスからシステムへデータのアップロードとシステムからデバイスへデータのプッシュはすべて自動的に行う．

3.2　　利用シーン

本システムを利用することで，想定する利用シーンについて述べる．

・　アドレス帳共有の利用シーンを図 3-2 に示す．

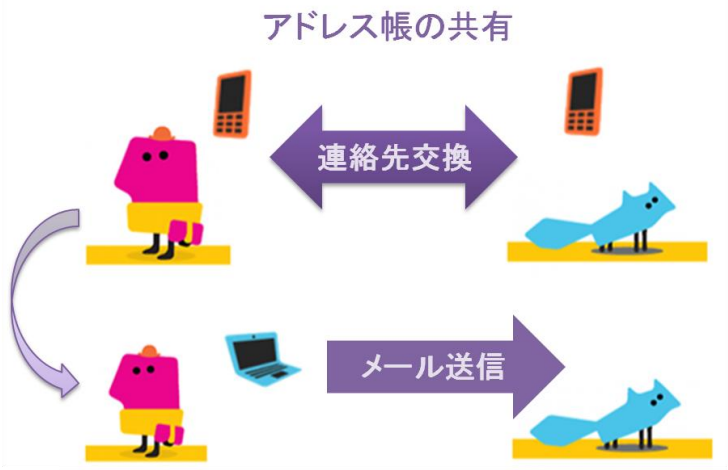


図 3-2　アドレス帳共有の利用シーン

携帯で交換した連絡先は更新された瞬間でシステムへ保存され，手動で同期する必要なく PC からメール送信が可能となる．

・　履歴共有の利用シーンを図 3-3 に示す．

図 3-3　履歴共有の利用シーン

電車中あるいは歩きながら閲覧したウェブページの履歴も，手動で同期する必要なくタブレット PC でも閲覧し続けできる．

3.3　関連システム

- iCloud[4]

アップル株式会社が開発した自社製品向けクラウドサービスである．

音楽，写真，書類などを自動で保存して，ユーザが使っているすべてのデバイス（iPhone, iPod，Mac etc…）にワイヤレスで同期するシステムである．

3.4　　共有するデータの範囲

本システムにおいて同期するデータの選択基準を以下に示す.

・MeeGo 既存アプリケーションが取り扱っているデータを利用する

本システムの開発において，開発期間の制限があるため，新たのアプリケーションを開発する変わりに，既存アプリケーションのデータをベースにした開発方式を選定した.

そのため，MeeGo 既存アプリケーションの内，普段よく使われ，またデバイス間の同期ニーズも高い点から，アドレス帳，ブラウザの閲覧履歴，メモとカレンダーを本システムの共有するデータの範囲にした.

3.5　　処理するデータの粒度

本システムのサーバーソフトウェアとクライアントソフトウェア内部で，処理するデータの粒度について，チーム内で以下の二つレベルについて検討した.

- 詳細レベル
  アドレス帳の共有を例とすると，連絡先一人に対して，電話番号，メールアドレスなど複数の項目は，それぞれ分けて前回変更時間などの情報を収集して管理する．また，データの更新も項目単位で行う.
- 概要レベル
  アドレス帳の共有を例とすると，連絡先単位で処理を行う.

また，それぞれのメリットとデメリットを表 3-1 に示す.

表 3.1　データ粒度

粒度	メリット	デメリット
詳細レベル	同時編集は可能	設計は困難 機器への負担は高い
概要レベル	設計は容易 機器への負担は低い	同時編集は不可

本システムを利用するユーザが所有する複数デバイスを同時に同じ項目（例えば，同じ連絡先）を編集することはあまりないため，データ処理の粒度は概要レベルを選定した.

3.6　　システム要件

3.6.1　　機能要件

データ共有システムの機能要件を以下に示す.

- ユーザ情報の管理
  ユーザが本システムを利用する時, ユーザ識別が必要となるため, ユーザの新規登録,

ユーザ情報の変更（パスワードなど）とユーザ削除の処理が必要となる．

・ デバイス情報の管理

本システムはデバイス間のデータ共有を行うため，デバイスの識別，デバイスの状態（オンライン，オフライン）など，デバイス情報の管理も必要となる．また，どのユーザがどのデバイスを所有しているなどの関連情報の管理も必要となる．

・ 対象データの監視・取得・更新

デバイスの対象データを監視し，ユーザがデバイス上で操作を行った場合（例えば：アドレス帳の追加や編集，ウェブブラウザでウェブページの閲覧 etc…），更新したデータの取得，サーバーへの送信が必要となる．

また，クライアントはサーバーから受信したデータに対して，機器に更新する必要もある．

・ クライアントとサーバーとの通信

本システムはクライアントとクライアントとの直接の通信に代わって，すべての通信はサーバーを経由する．

・ データの解析・分析・処理

本システムのコア機能である．デバイス間のデータ共有を行うため，データの解析，分析などの処理が必要となる．

3.6.2 非機能要件

要件定義フェーズでは「個人情報の取り扱い」,「無線ネットワークの通信環境」,「対象デバイスへの負担」,「将来の拡張」の四つ観点から本システムの非機能要件を洗い出した．以下にそれぞれ述べる．

・ セキュリティ

本システムはアドレス帳，カレンダー，メモ帳などの個人情報データをネットワーク経由して送受信を行うため,通信時は暗号化(SSL)を行い盗聴されないように留意する必要がある．

サーバー上の別のユーザの情報に誤ってアクセスしないよう，通信時にはユーザ認証を行う必要がある. また, ユーザ ID, パスワードと共有データ自体は暗号化して保管し，システム管理者側からも見えないようにする．

データは全て DB に保存し，ユーザごとに DB を切り分けておくことでユーザのデータに他のユーザからアクセスできないようにする．

そのため，以下の二つ方面を暗号化する必要がある．

- 通信の暗号化

  サーバーとクライアントの通信はすべて暗号化する．

- データベースの暗号化

  サーバー管理者でもユーザの個人情報を見られないようにする．

・ 信頼性

クライアントデバイスのネットワーク環境は無線ネットワークを想定する前提で，不安定の接続でも正確的に処理する必要がある．

各ファイル・履歴情報が最新のものになるように留意する必要がある．また，途中で通信が切断された場合でも，各ファイルを破壊せず通信が終わったファイルだけ更新できるようにしなければならない．

・ 効率性

クライアントソフトウェアが常に機器のデータを監視しても，他のプログラムや処理を阻害してはいけない．

また，容量の大きいファイルは分割して送信するように，ユーザの通信を阻害しないようにする必要がある．

・ 拡張性・移植性

将来的に共有データの追加，他種類のクライアントデバイスの対応（Meego 以外の端末）も容易である設計が必要となる．

3.7 システム構成

システムの構成を図 3-4 に示す．

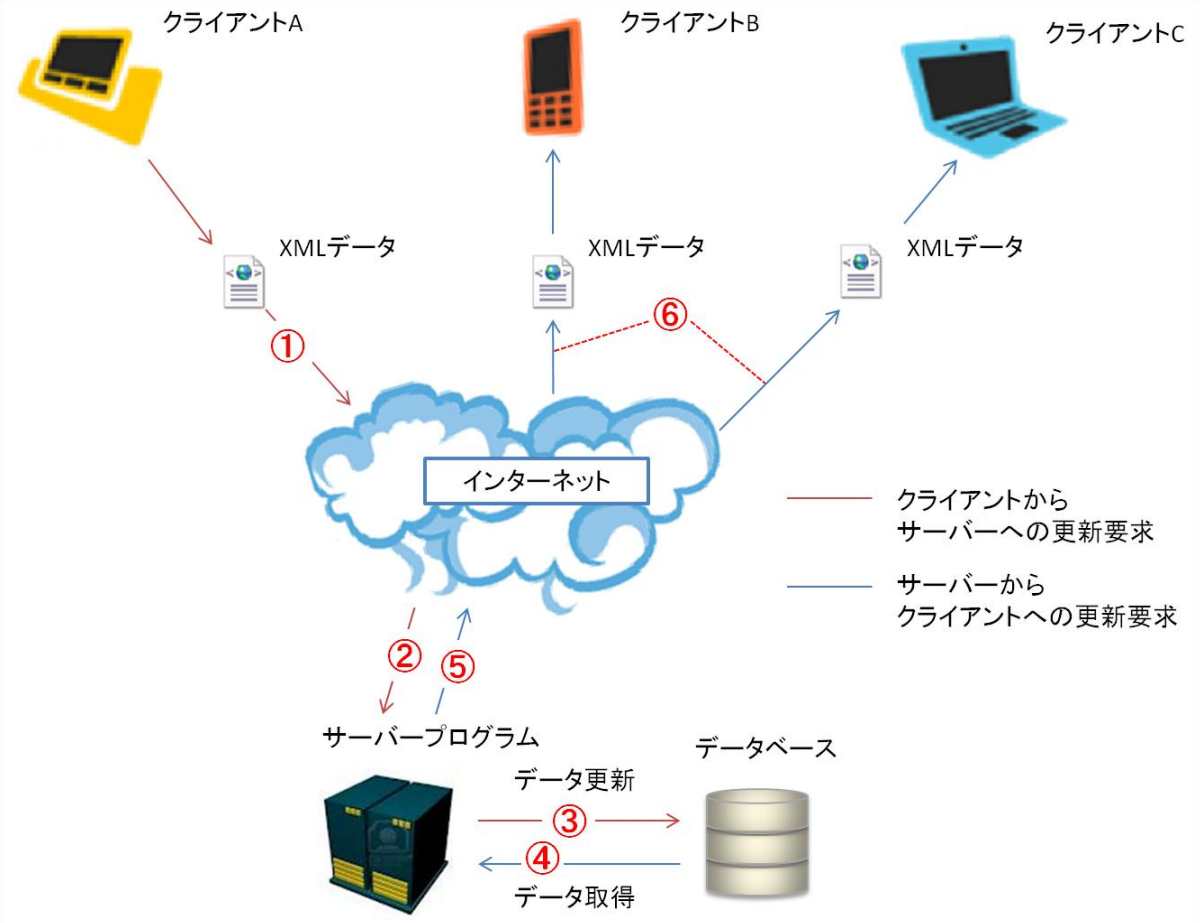


図 3-4 システム構成図

本システムデータ共有の流れは赤字部分に示した通り，自動的に行う．各個デバイスとサーバーはインターネットを経由し，サーバーはデバイスからの XML 形式の更新要求（新規追加，変更，削除）とデータを受け取って処理し，結果を他のクライアントへ送信する．クライアントはサーバーから受け取った更新要求とデータに対して，反映をおこなう．また，すべてのクライアントのデータはデータベースに保存する．

3.7.1 制約条件

本システムが利用する対象デバイスは MeeGo タブレットシステム，MeeGo ノートブック

システムがインストールされた機器である．
　また，本システムはインターネット経由でデータを送受信するため，インターネットに接続できないデバイスは，本システムの利用対象外とする．

3.7.2　　通信方式

本システムの通信方式を表 3-2 に示す．

表 3.2　通信方式

通信方式	ソケット通信
通信データ形式	XML 形式

3.7.3　　ソフトウェア環境

本システムのソフトウェア環境を表 3-3 に示す．

表 3.3　ソフトウェア環境

種類		名称	バージョン
サーバー	OS	Windows Server2008	R2 64Bit
	OS（開発用）	Windows7 Professional	32Bit
	IDE	Visual Studio 2010	Express Edition
	データベース	MySql	5.5
	暗号化フレームワーク	OpenSSL.NET	0.5 RC1 64Bit
	開発言語	Visual C#	
クライアント	OS	Meego Tablet/ MeeGo Notebook	1.2
	OS（開発用）	Ubuntu Desktop	10.04
	IDE	QT Creator	2.1

	SDK	MeeGo SDK	1.2
	暗号化フレームワーク	OpenSSL	1.0.0e
	開発言語	C++	

第4章　　機能細分と担当した機能

本システムの開発フェーズで，機能を細分したモジュール構成を図 4-1 に示す.

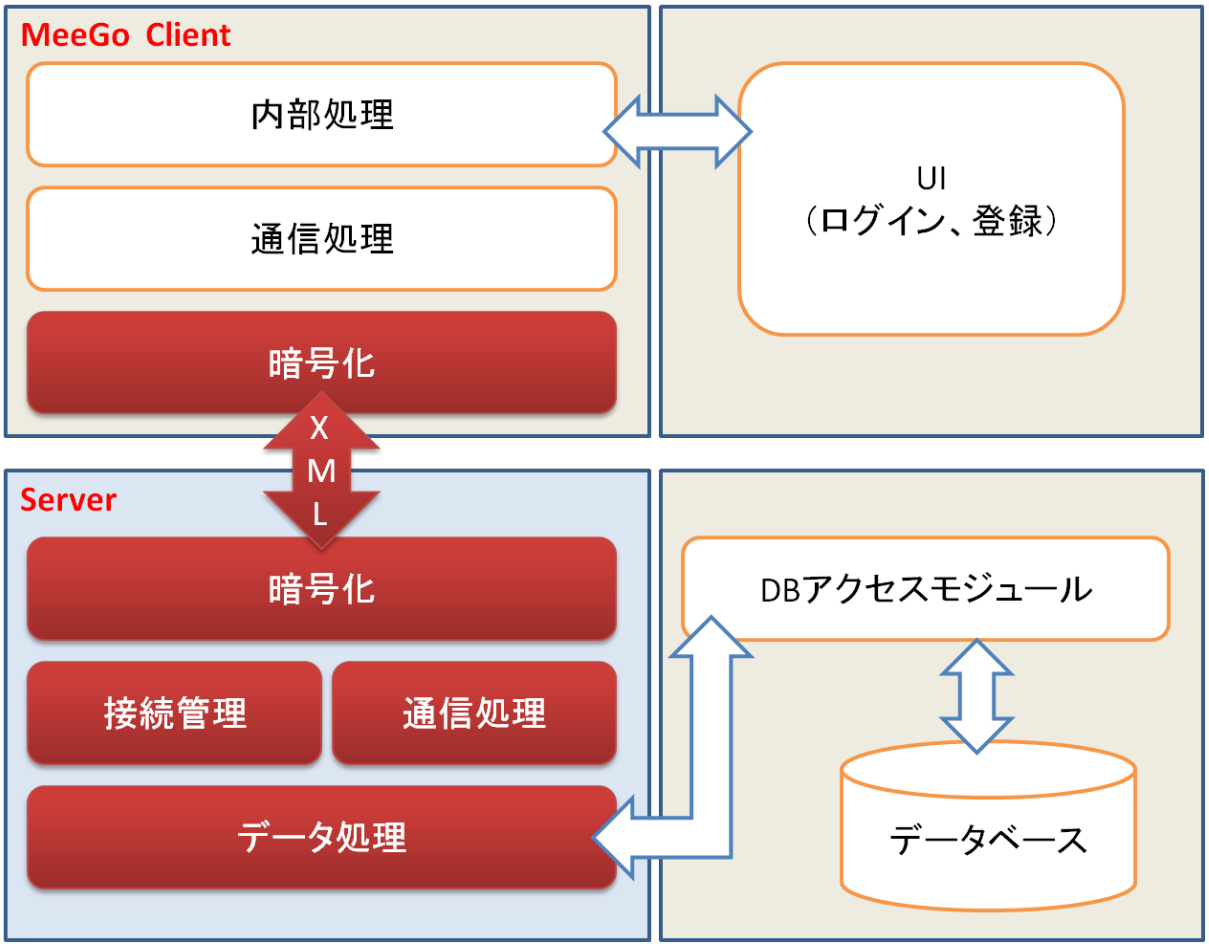


図 4-1　　詳細モジュール

クライアント側では，以下の四つのモジュールから構成される.

1. UI モジュール

画面操作結果を受け取るモジュールである．ユーザの新規追加，ユーザ情報の更新，処理結果メッセージの表示などの機能がある.

2. 内部処理モジュール

デバイス中のデータの監視，変更やデータの比較処理などを行うモジュールである.

3. 通信処理

サーバーとのデータの送受信を行う通信モジュールである.

4. 暗号化モジュール

サーバーへ送信データの暗号化とサーバーから受信したデータの復号化を行うモジュールである.

サーバー側では，以下の五つのモジュールから構成される.

1. 暗号化モジュール

クライアントへ送信データの暗号化とクライアントから受信したデータの復号化を行うモジュールである．

2. 接続管理モジュール

複数のクライアントの接続情報を管理するモジュールである．接続の管理，サーバーリソースの管理，デバイス間の通信などの機能がある．

3. 通信モジュール

デバイスとの通信を行うモジュールである．

4. データ処理モジュール

デバイスから受信したデータの解析，処理を行うモジュールである．

5. DB アクセスモジュール

データベース操作を処理単位で纏めて，ライブラリとしてデータ処理モジュールに提供するモジュールである．

上記モジュールの内，著者は図の赤色部分であるサーバーソフトウェア，通信用 XML データ，システムの暗号化機能の設計開発を担当した．

第5章　データ共有システムのサーバーソフトウェアの設計及び開発

本章では著者が担当した部分であるサーバーソフトウェア構成と実装した各個処理モジュールについて述べる.

5.1　開発環境について

サーバーソフトウェアの開発環境について，Visual Studio を用い Visual C#で開発をおこなうか，若しくは，クライアント側の環境と同様に MeeGoSDK を用い C++で開発をおこなうかについて検討した.

本システムのサーバーとクライアントとの通信方式はソケット通信，また送受信するデータの形式は XML であるため，ともに開発フレームワークと開発言語に依存性はないと判明した．また，サーバーソフトウェアの開発担当者（著者）の開発経験より，Visual C#の経験は多いため, Visual Studio を用いた Visual C#での開発は MeeGoSDK を用いた C++での開発より短時間で開発できると考えたため, 開発環境は Visual Studio を用いた Visual C#での開発を選定した.

さらに，本システムのサーバー構成は，Visual Studio の開発環境に合わせるため，Windows サーバーを選定した.

5.2　サーバーソフトウェアにおける同期処理の流れ

サーバーソフトウェア同期処理の流れを図 5-1 に示す.

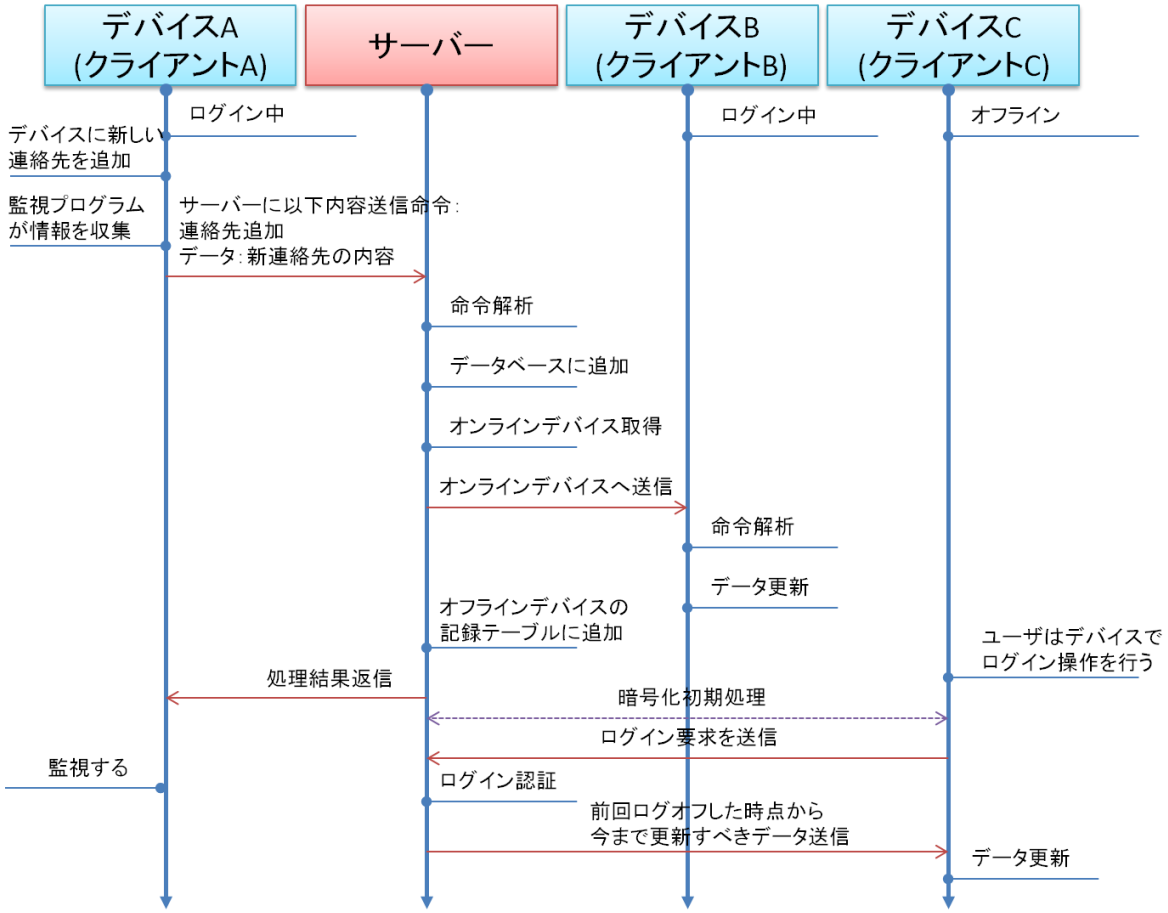


図 5-1　サーバーソフトウェア処理の流れ

図 5-1 の接続中のデバイス（クライアント）は三つの場合の例である.

初期状態として，クライアント A とクライアント B はオンライン（ログイン中）で，クライアント C はオフラインの状態とした.

サーバーはクライアント A からのデータを解析，DB アクセスモジュールを用いてデータベースに追加する.その後,接続管理モジュールはオンラインにするデバイス情報を取得し，データ送信する.（デバイス B にデータを送信する）

クライアント C の接続要求が来た場合，まず暗号化の初期処理を行う，その後クライアントからのログイン情報を認証する．ログインが成功した直後，サーバーソフトウェアは今回追加されたデータを含め，前回ログオフした時点から更新すべきデータをクライアントに送信する.

5.3　クラス設計

サーバーソフトウェアのクラス構成を図 5-2 に示す.

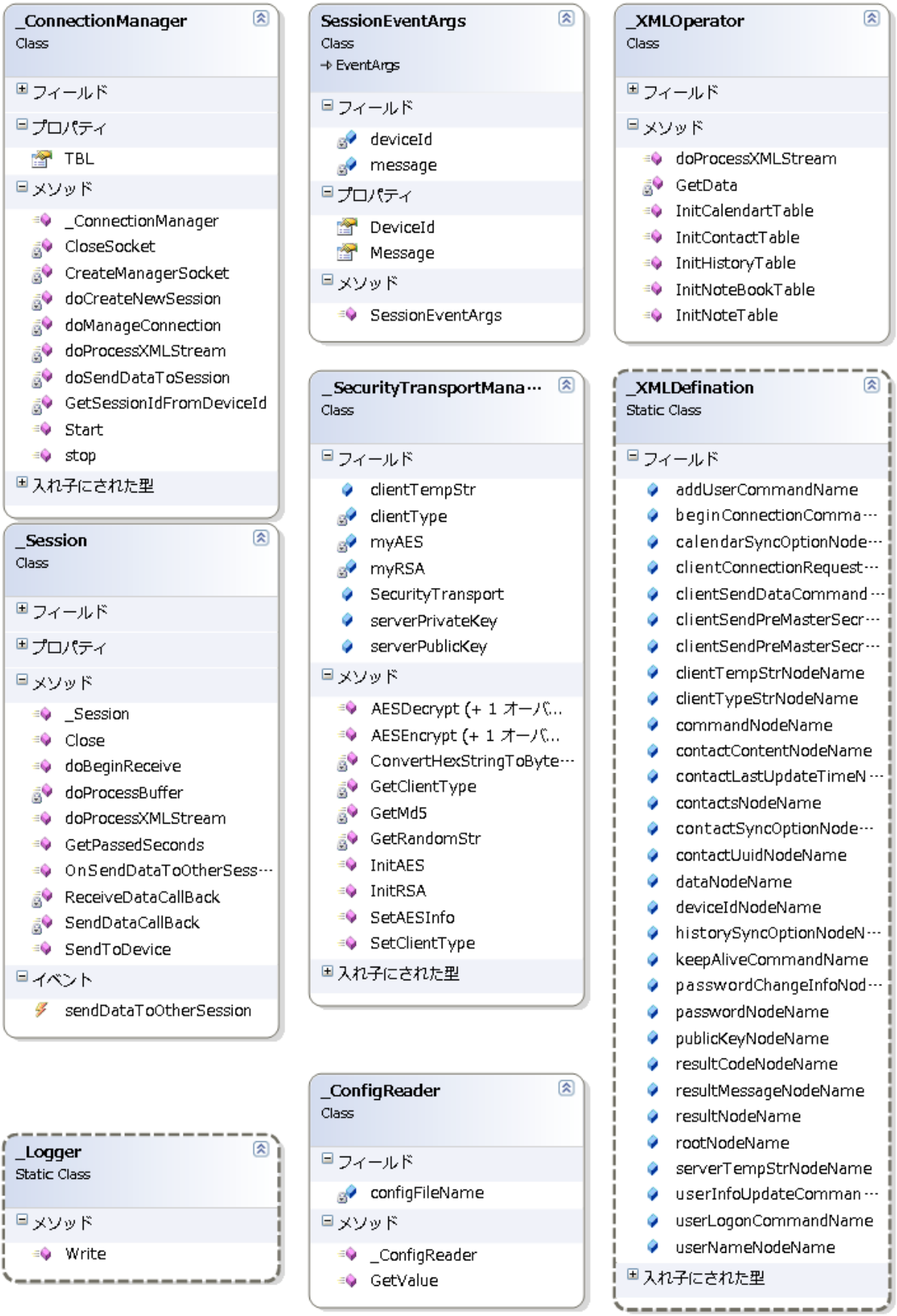


図 5-2　サーバーソフトウェアクラス構成

- _ConnectionManager クラスはサーバーソフトウェアのコアクラスである．セッションの作成，接続状態の管理やセッション間の通信などを行う．
- _Logger クラスはシステムログ記録用クラスである．
- _Session クラスはデバイスとの送受信を処理するクラスである．_Session クラスのインスタンスとデバイスの関係は一対一である．
- SessionEventArgs クラスは，_ Session クラスが利用するイベントクラスである．セッション間通信の際に使う．
- _SecurityTransportManager クラスは通信のセキュリティを管理するクラスである．セキュリティの初期化処理とデータの暗号・復号などを行う．
- _ConfigReader クラスはデータベースの IP アドレスやソケットポットなどを記載した INI ファイルを読み込む専用クラスである．
- _XMLOperator クラスは，XML データの作成・解析・処理などを行うクラスである．
- _XMLDefination クラスは, XML ノードの名前などの定義情報を保存するクラスである．

5.4　　画面設計

サーバーソフトウェアの画面は以下の二つの部分からなる．

- タスクバー画面
- モニタリング画面

サーバーソフトウェアの画面イメージを図 5-3 に示す．

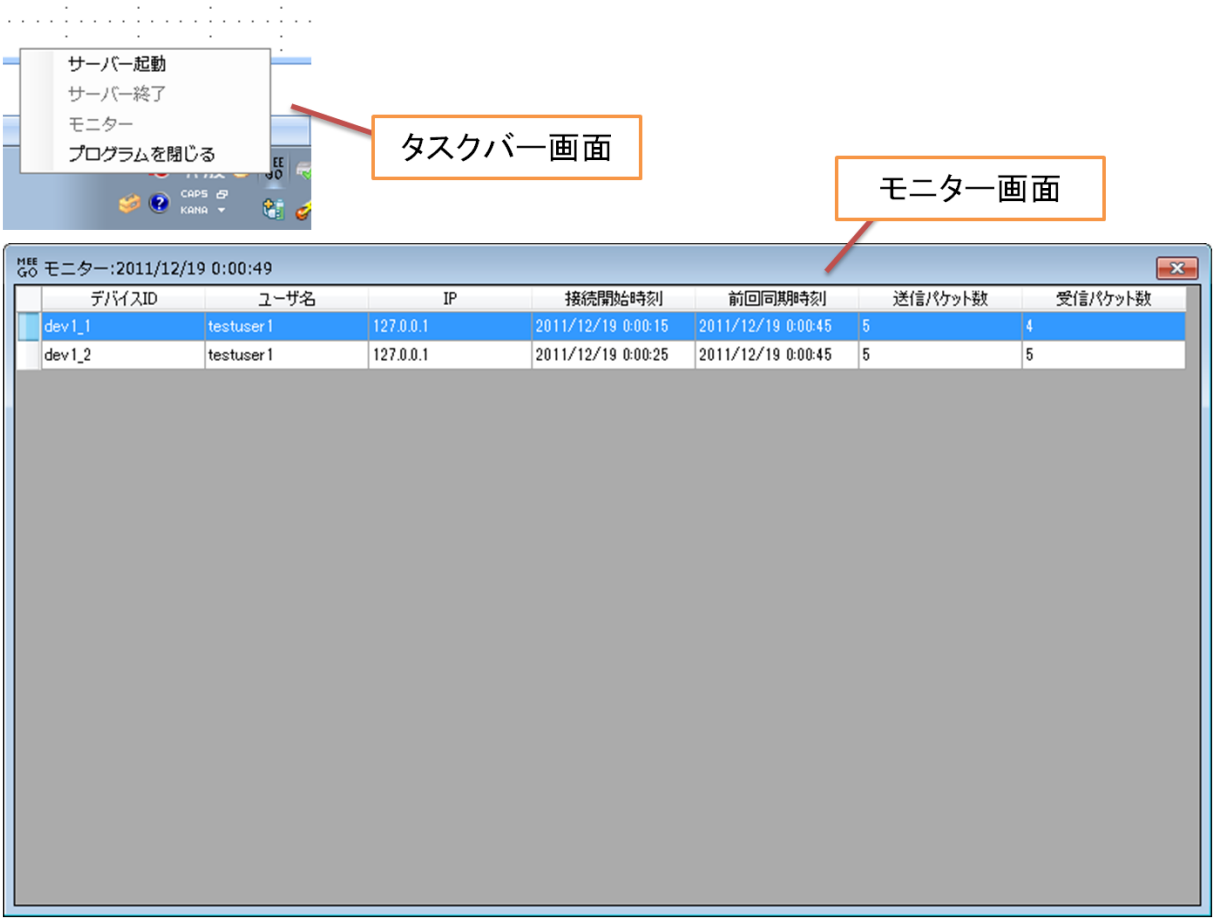


図 5-3　　プログラム画面

タスクバー画面には，サーバーの起動・終了，接続のモニタリングとプログラムを閉じる四つのボタンからなる．

また，サーバー起動中の場合，タスクバー画面のモニターボタンが有効になる．モニターボタンを押下すると，モニタリング画面が表示される．

モニタリング画面には，現在接続中のデバイス情報を一秒ごとに更新して表示する．画面に表示する項目を以下に示す．

- デバイス ID
  該当デバイスの MAC アドレス
- ユーザ名
  該当デバイスを所有するユーザの名前
- IP
  デバイスの IP アドレス
- 接続開始時刻
  接続開始した時間
- 前回同期時刻
  前回データを同期した時間
- 送信したパケット数
  サーバーから該当デバイスに送信したパケット数
- 受信したパケット数
  該当デバイスからサーバーに送信したパケット数

5.5 サーバーソフトウェアのモジュール構成

システムの機能要件と非機能要件に合わせて，著者が設計したサーバーソフトウェアの位置付けと細分化したモジュール構成を図 5-4 に示す．

図 5-4　サーバーソフトウェア構成

また，データベースとのやり取りは，著者が基本設計を担当した DB アクセスモジュールが提供する操作 API で行う．

5.5.1　通信処理

本節では，サーバーソフトウェアの通信モジュールの設計と実装の詳細について述べる．

- ソケット通信[5]

サーバーソフトウェアとクライアントソフトウェアとの通信はソケット方式を採用した．ソケット通信の流れを図 5-5 に示す．

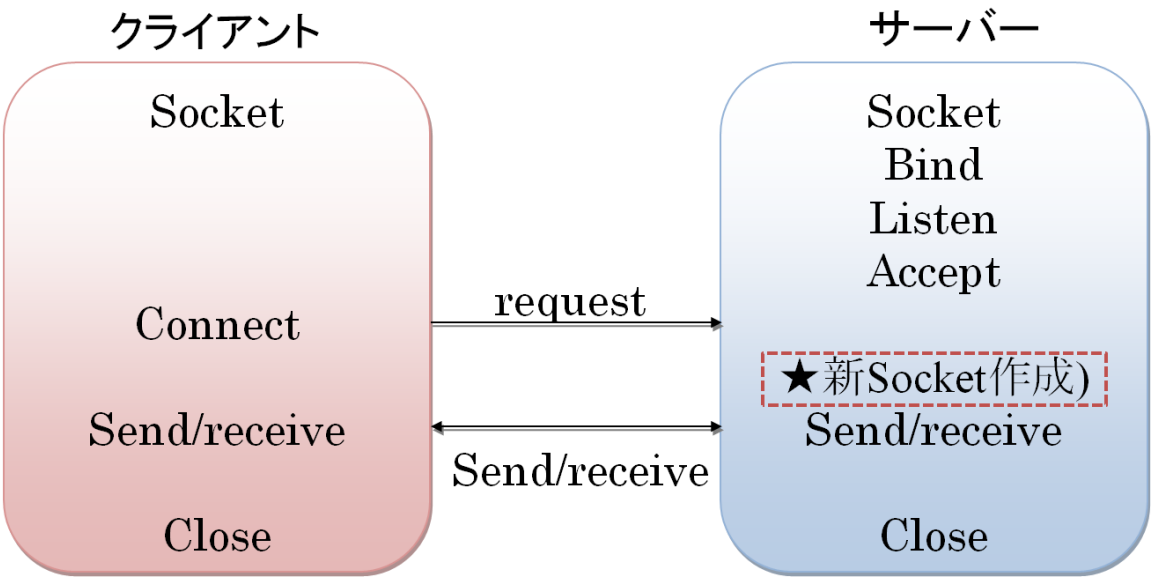


図 5-5　ソケット通信

ソケット通信の初期処理の流れを以下に示す.

- サーバーソフトウェアはソケットを作成する
- サーバーソフトウェアはソケットとポートの結合（Bind）をおこなう
- サーバーソケットは接続キューを作成（Listen）する
- サーバーはクライアントからの要求を待つ（Accept）
- クライアントはソケットを作成する
- クライアントソケットはサーバーソケットに接続（Connect）する
- サーバーは接続を受けて，この接続の処理を受けるソケットを作成する
- サーバーとクライアントはデータの送受信を行う（Send/Receive）

● 非同期ソケット通信[6]の実装

標準のソケット通信方式で実装した場合，サーバーソフトウェアはクライアントからのデータを受信したまで，待つ状態になり，続く処理ができなくなる．この方式のソケット通信は，同期ソケット通信と言われる．.NET Framework デフォルトのソケット通信方式は同期ソケット通信方式である．

また，同期ソケット通信に対して，非同期ソケット通信というデータの受信処理を待たずに続けて処理できる方式もある．

非同期ソケット通信を実現するため，以下の二つの方法を挙げられた．

- 新たにスレッドを作って，同期処理を行うことで，親スレッドは続いて処理できる．
- .NET Framework が用意された非同期方法[7]（BeginSend/BeginReceive など）を実装する．

そのうち，新スレッド方式の実装は簡単であるが，スレッドの数量が一定以上になると，システムのパフォーマンスが低下する．本サーバーソフトウェアは，パフォーマンスを重視するため，.NET Framework が用意された非同期方法を実装した．

5.5.2　接続管理

本システムサーバーソフトウェアの接続管理は，.NET Framework が提供されたスレッド，I/O スレッド[8]と連想配列などの要素で，接続管理スレッド，セッションオブジェクト連想

配列（Dictionary）とセッションオブジェクトから構成した．

・　接続管理スレッド
　サーバーサービス起動後常に実行する常駐スレッドである．接続管理スレッドは 5.5.1 項にて説明したサーバーソケットを実装する．新たなデバイスからの接続（ソケット）要求に対して，セッションオブジェクトを作成し，セッションオブジェクト連想配列に追加する．また，セッションのログアウト，セッションタイムアウトや他の原因で通信中断してしまった場合，セッションオブジェクト連想配列に保存されたセッションオブジェクトのリソースを解散し，連想配列から削除する．
　また，デバイスのオンライン・オフライン情報はデータベースに保存する．
・　セッションオブジェクト連想配列（Dictionary）
　.NET Framework の Dictionary クラスのインスタンスである．中身は key/value のようなデータを保存される．Key 情報は 1 から重複しない整数で，value はセッションオブジェクトである．接続管理スレッドは key 情報でセッションオブジェクトのインスタンスを取得して，処理を行う．
・　セッションオブジェクト
　クライアントとの通信（非同期ソケットを実装する），暗号化，XML データの解析や処理などすべてのデバイスとのやり取りを行うデバイスと一対一作成されたオブジェクトである．

● デバイスのログイン・ログアウト
デバイスがログイン，ログアウトする時の接続管理の構成を図 5-6 に示す．

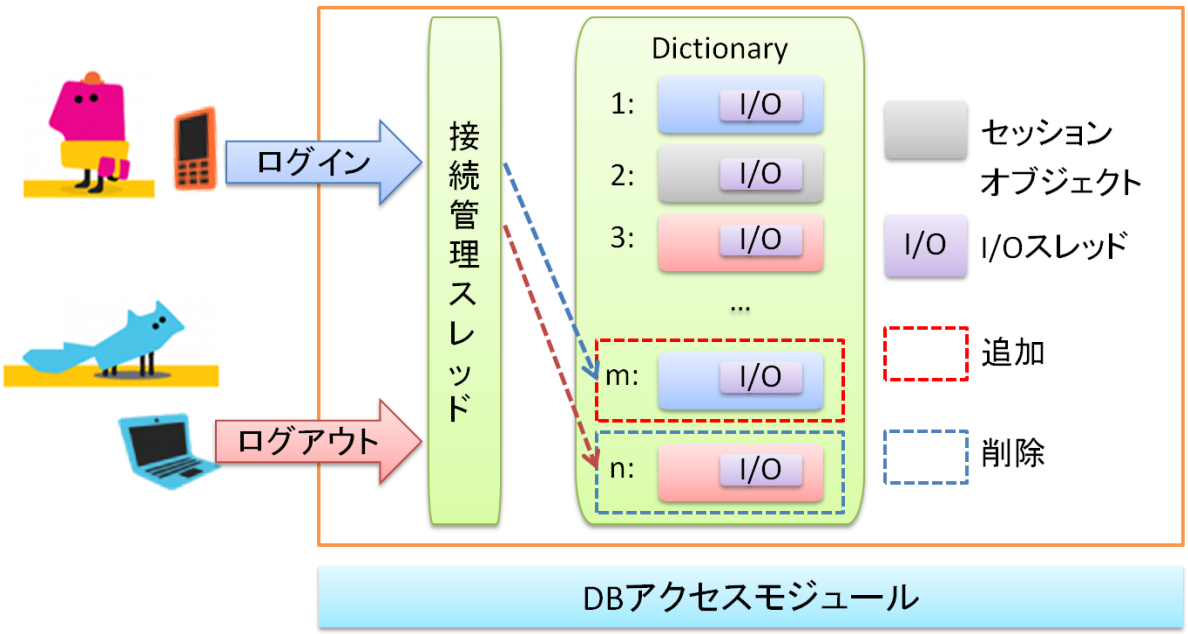


図 5-6　クライアントログイン・ログアウト

　ログインデバイスに対して，接続管理スレッドはセッションオブジェクトを作成し，セッションオブジェクト連想配列に追加する．
　また，ログアウトデバイスに対して，接続管理スレッドは使わないオブジェクトを削除する．

● セッションオブジェクト間の通信

セッションオブジェクトはデバイスから取得したデータを他のデバイスへ送信する必要がある．また，他のデバイスへの送信はそのデバイス専用のオブジェクトで行うため，オブジェクト間のデータ通信が必要になる．

セッションオブジェクト間の通信を図 5-7 に示す．

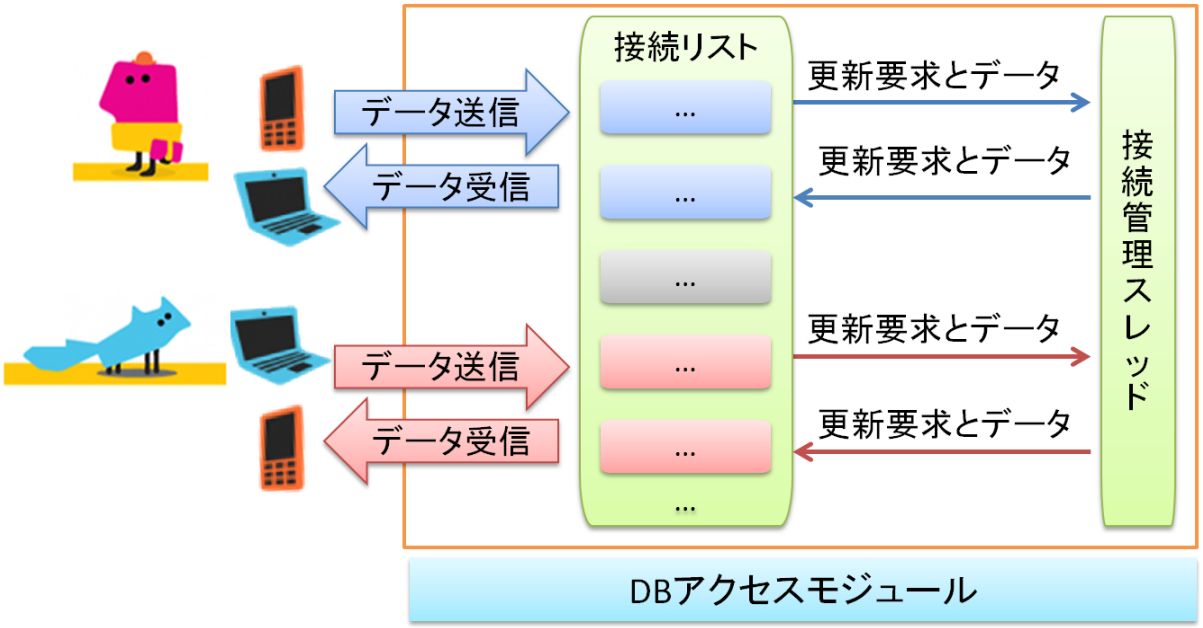


図 5-7　セッションオブジェクト間の通信

図 5-7 は，二つのデバイスが同時にデータ更新を行って，サーバーソフトウェアが処理する場合の例である．処理流れは同じであるため，上のデバイス（青い部分）の処理を説明する．

セッションオブジェクトはまず DB アスセスモジュールを利用し，現在オンラインである他のデバイス情報を取得し，接続管理スレッドに送信する．接続管理スレッドはデバイス情報でセッションオブジェクトリストからセッションオブジェクトを検索して，オブジェクトにデータを送る．最後に，更新データを貰ったセッションオブジェクトは自身が取り扱うデバイスにデータを送信する．

5.5.3　XML データ処理

本項ではサーバーソフトウェアが行う XML データの解析・生成・処理について述べる．

● 本システムが取り扱う XML データの基本構造を表 5-1 に示す．

XML データはルートノードと三つのサブノードからなる．

ルートノードとコマンドノード，データノード，処理結果ノードの名前は固定である．

コマンドノードは必須ノードで，コマンドは文字列でサブノードを持たないとする．

データノードはコマンドノードによって省略可能で，サブノード数の制限はない．また，サブノードの定義はコマンドによって異なる．

処理結果ノードは省略可能で，クライアントかサーバーは受信したデータのコマンドを処理してから，結果通知データを格納するノードである．

また，XML データには日本語が含まれるため，XML データの文字コードは“UTF8”である．

表 5.1　XML データの構造

ルートノード	MeegoSync
コマンドノード	Command
コマンドのデータノード	Data
処理結果ノード	Result

サーバーソフトウェアが処理する XML データの例を図 5-8 に示す．

```
<?xml version="1.0" encoding="utf-8"?>
<MeegoSync>
 <Command>ClientSendData</Command>
 <Data>
  <Contact>
   <Uuid>92263505</Uuid>
   <Content>Mekkk,,,,,,,,,,,,</Content>
   <LastUpdateTime>2011/12/16 1:44:49</LastUpdateTime>
  </Contact>
  <Contact>
   <Uuid>249626337</Uuid>
   <Content>AFFFF,AFFFF,,666666,,,,,,,,,</Content>
   <LastUpdateTime>2011/12/16 1:45:14</LastUpdateTime>
  </Contact>
 </Data>
</MeegoSync>
```

図 5-8　XML データ例

以上が，クライアントからサーバーへアドレス帳二つを送信する時の XML データである．

- XML データの変換・解析

本システムが処理する XML データは基本の XML 形式であるため，XML の変換や解析の実装は.NET Framework が提供する System.Xml 命名空間の XmlNode クラスと XmlDocument クラスで行った．

また，データノードのサブノード数と形式は固定でないため，XML データの解析を行う

時に，コマンドノードを先に行うようにした．
解析したコマンドノードのデータは文字列である．また，データノードと結果ノードは連想配列（Dictionary）を利用し，key/value の形で保存する．

- XML データの処理

XML データの処理フローを図 5-9 に示す．

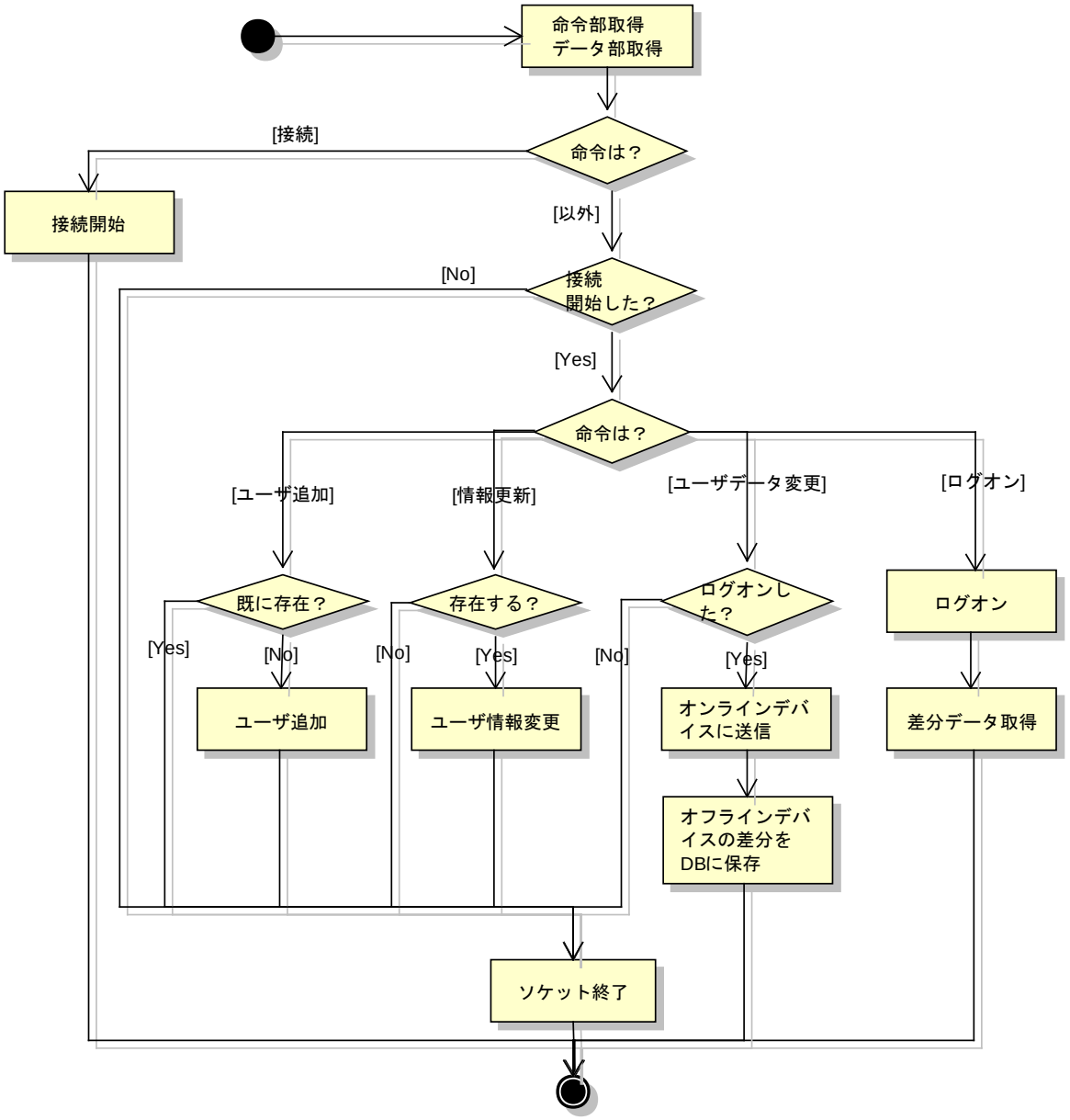


図 5-9　XML データの処理フロー

クライアントからの XML データは，「接続-ログイン-データ送受信」の流れである．安全性やサーバーへの負担を考慮した結果，ユーザ追加後，ユーザ情報を変更後や不正データと判明した場合は直ちにセッションを切る（ソケット終了する）ように設計した．

5.5.4　　分割受信及びデーター貫性チェック機能

本システムのクライアントのハードウェア制限と無線ネットワーク状況の不安定性により，サーバーへ大量にデータが送信されると，クライアント側の処理時間が長くなり，送信失敗率が高くなる．そのため，クライアント側では状況によって，大量のデータを小さいサイズのデータに分割してから送信するようにした．

また，データのサイズと関係なく，ネットワークでのデータ転送は，無事受信しても，なにかの原因でデータの中身が壊れている可能性も考えられる．

サーバーソフトウェアでは，以上の二つの問題に対して，設計・実装を行った．

分割受信及びデーター貫性チェックのイメージを図 5-10 に示す．

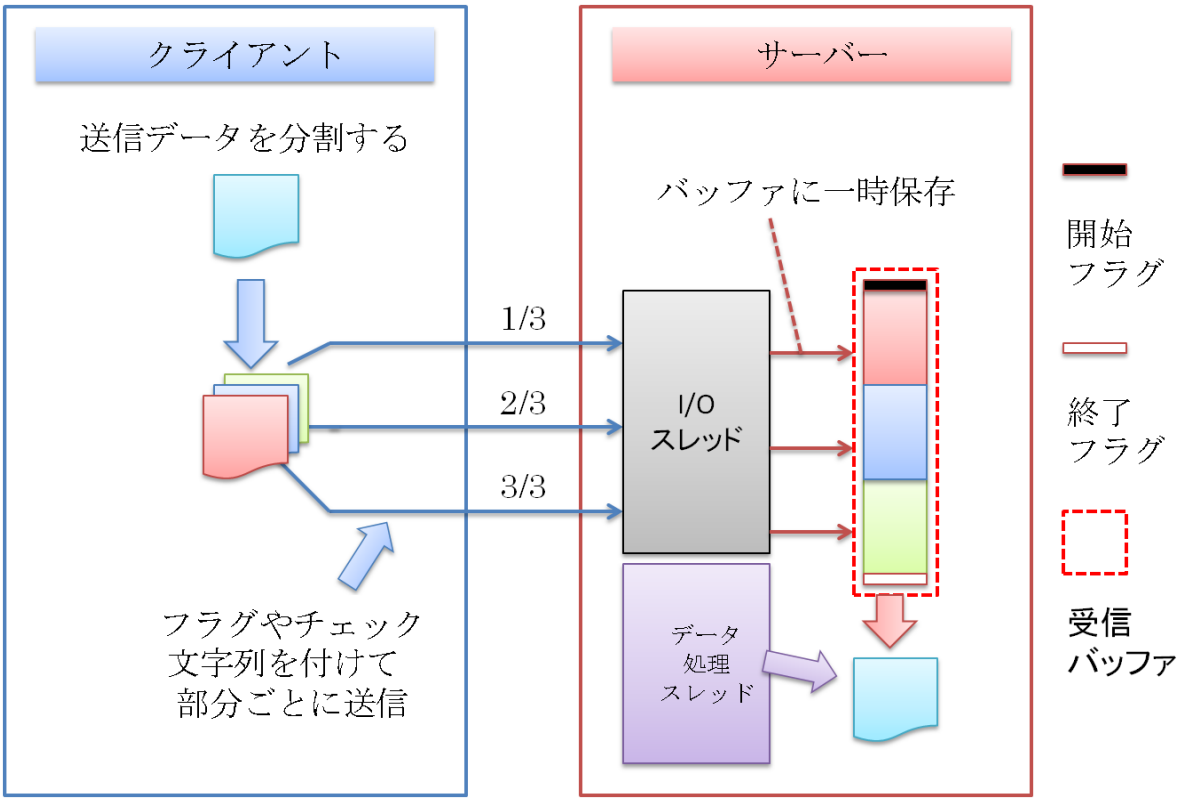


図 5-10　　分割受信及びデーター貫性チェック

設計の経緯を以下に示す．

分割受信：

普段の場合は，I/O スレッド（非同期ソケット）の受信メソッドは，１回受信した後，すぐに XML の解析処理を行って，命令を取得し，データ処理を行う．しかし，分割受信の場合，１回受信したデータは，一つの XML ストリームか，または XML ストリームの一部であるかは判別できない．

そのため，XML ストリームの頭と後ろにストリームの開始フラグと終了フラグを付ける設計にした．それに対して，以下の二つの問題があった．

- 開始フラグと終了フラグの選定
  開始フラグと終了フラグは XML ストリーム本文に含まれると，混乱になる．
- 分割データの保存

まず，開始フラグと終了フラグの選定について，元の XML データは文字列から構成されるため，文字でないものをいくつか候補に選んだ．そのうち ASCII コード１（SOH）と ASCII

コード１２７（DEL）を開始フラグと終了フラグとした．

また，サーバーソフトウェアは，分割された部分データをバッファに保存し，一つの XML ストリームをすべて受信したら，データ処理スレッドの処理待ち行列に追加する．

データ一貫性チェック：

一貫性チェックについて，チェック内容によって，チェックの項目が複雑になると，受信したデータの正確性が高くなるが，チェック用データの量は多くなり，逆に通信の阻害になる．そのため，サーバーとクライアント両方への負担をあまりかけないために，本システムでは文字サイズのチェックのみを行う．

クライアントから送信する時，XML 本文の前に，送信しようとするデータの UTF8 モードのバイト数を付ける．サーバーソフトウェアは，受信したデータを解析し，データの中に保存されたバイト数で，実際受信した文字数と比較する．

5.5.5 サーバープッシュ機能

本システムのサーバーはクライアントから受信したデータを他のクライアントへ送信する必要がある．サーバーからクライアントへ送信する時の実現方式について，以下の二つが考えられる．

- クライアントからサーバーへ更新メッセージを発送する
  クライアントは一定時間ごとにサーバーへデータ更新依頼のメッセージを発送する．サーバーはメッセージを受信した後，クライアントの更新すべきデータを取得してクライアントへ返信する．
- サーバーからクライアントへ直接データを送信（サーバープッシュ）する
  サーバーとクライアントは接続状態を保持する．サーバーはクライアントの更新すべきデータがある時に，直接クライアントに送信する．その結果，クライアントからサーバーへ更新依頼メッセージを送信する必要がなくなる．また，クライアントは随時サーバーからデータを取得できるため，同期の時間が減るとともに，ユーザ体験も向上できる．

本システムはユーザ体験を重視，同期処理を最短時間で完了させることを目指し，サーバープッシュ方式を選んだ．サーバープッシュ方式の構成を図 5-11 に示す．

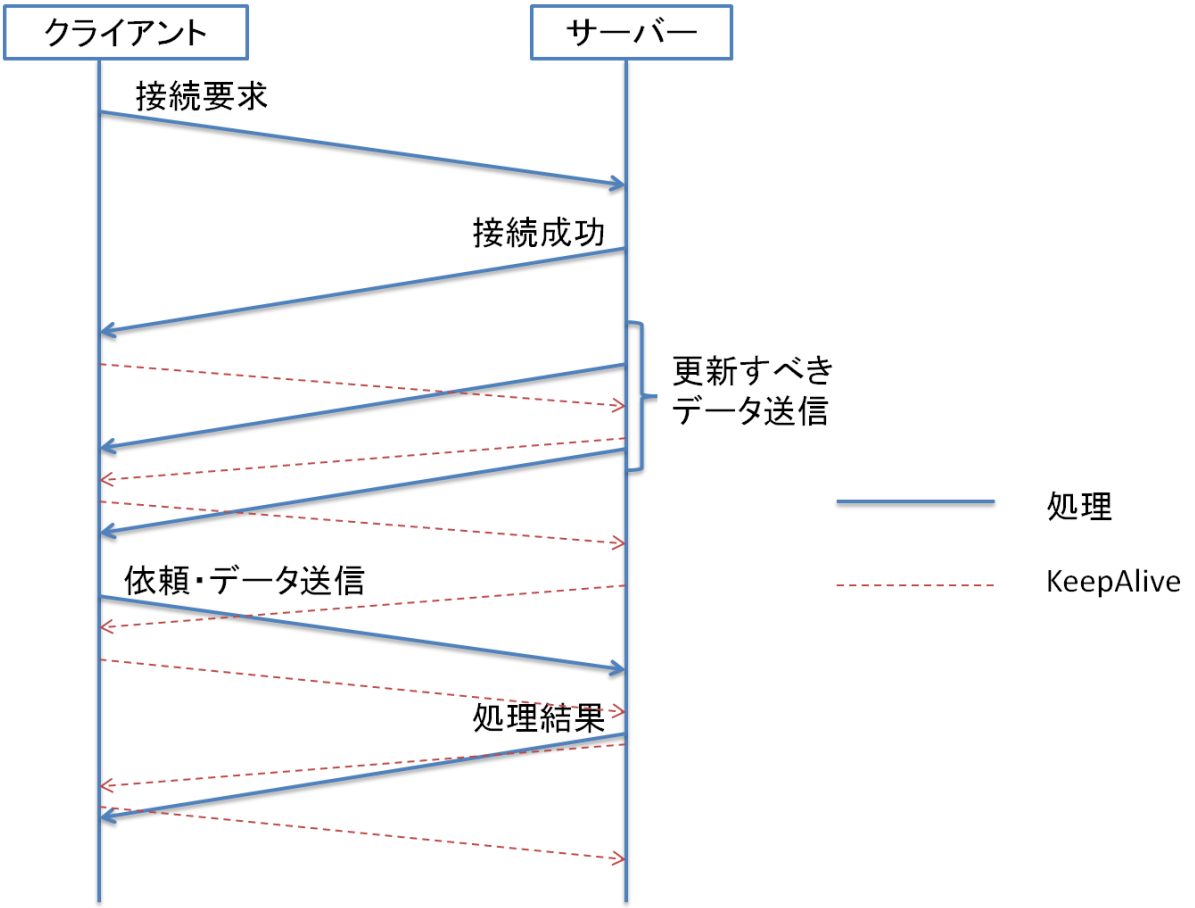


図 5-11　サーバープッシュ

赤色点線は，サーバーとクライアントの接続状態を保持するため，相手に送信する確認パケット（KeepAlive パケット）である．

まずクライアントからサーバーへ確認パケットを送信，サーバーは確認パケットを受信した後返信する．サーバーとクライアントは一定時間内相手の確認パケットが来ない時，接続問題が発生したとみて，接続を切るように実装した．

また，KeepAlive パケットの内容を図 5-12 に示す．

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>KeepAlive</Command>
</MeegoSync>
```

図 5-12　KeepAlive パケット

5.5.6　暗号化処理機能

本システムのサーバーとクライアントとのデータ通信を行う前に，まずはサーバー認証と送受信データの暗号化処理を行う．そうすることで，通信中のデータが第三者に流出しても，データを復号するための鍵がないため，データの内容を見られず，ユーザの個人情報が保護できる．（設計の経緯は節 6.2 で述べる）

サーバー側で行う暗号化処理の流れを図 5-13 に示す.

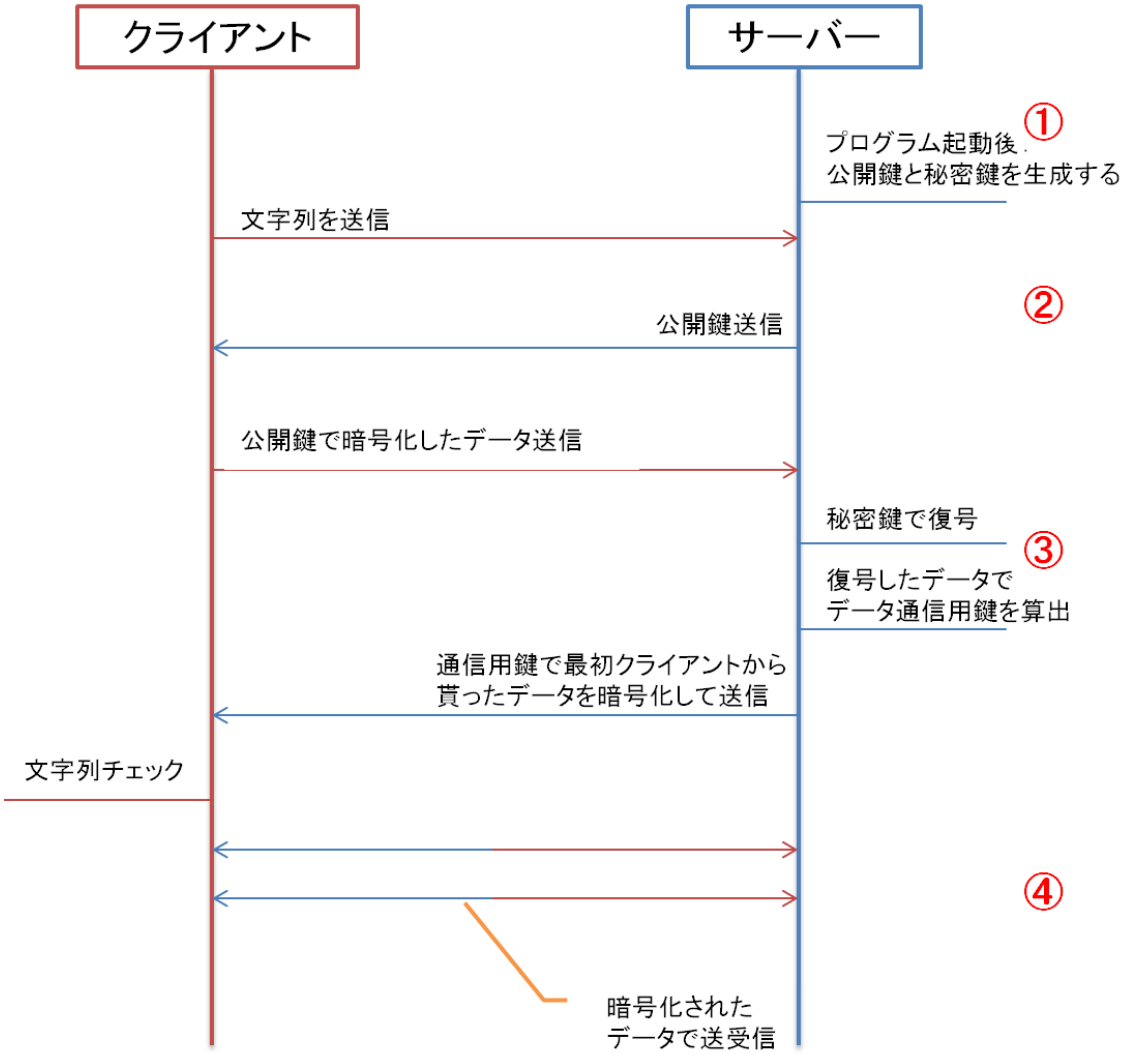


図 5-13　サーバー側で行う暗号化処理

サーバー側の暗号化処理は以下の四つの段階からなる.

1. プログラム起動後，公開鍵と秘密鍵を生成する.
2. 新たのクライアントからの接続要求に対して，公開鍵を送信する.
3. クライアントから受信したデータを秘密鍵で復号化して，データ通信用対称鍵を算出し，最初クライアントから受信した文字列を対称鍵で暗号化して，クライアントへ返信する.
4. クライアントからの返信を待つ．また，これからの送受信は対称鍵で暗号化・復号化を行う.

第6章　　他の担当した部分について

本章では第５章のサーバーソフトウェアの開発以外に，技術調査や設計開発フェーズで著者が担当した他の部分について述べる．

6.1　　技術調査

システムの設計開発を行う前に，著者が実現性の検証や技術手法の選定などで行った技術調査について述べる．

● クライアントソフトウェアの実現性調査

Meego にはすでにアドレス帳やカレンダーなどのアプリがプリインストールされているため，本システムの実装方針はこれらのプログラムをそのまま利用し，同期サービスを追加することである．

目標としたクライアント側のデータ更新の仕組みを図 6-1 に示す．

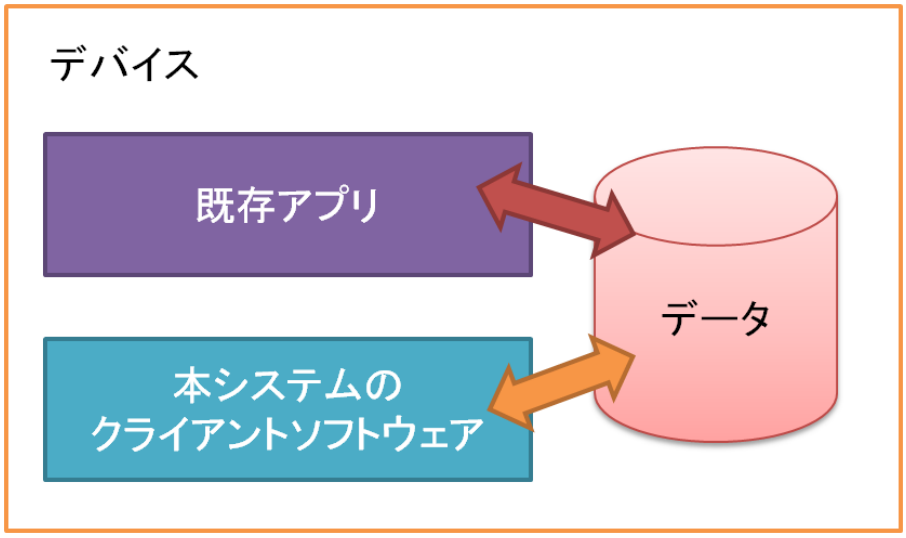


図 6-1　　データ更新の仕組み

既存アプリと本システムのクライアントソフトウェアは同じデータ対象にアクセスするため，本システムのクライアントソフトウェアは，既存アプリのデータアクセス方法を分析する必要がある．

著者は MeeGo バージョン 1.2 のタブレット版にプリインストールされたアドレス帳ソフトウェアを対象として調査を行った．ソースコード[9]を分析し，アドレス帳データの追加と取得するためのデモプログラムを作成した．

その結果，このような仕組みは実装可能と判別した．

● 既存同期フレームワークの再利用における調査提案

本システム技術調査フェーズでは，チーム内で，以下の二つの同期処理の実装方針について議論した．

・ Meego プラットフォーム既存のフレームワークを利用

MeeGo バージョン 1.2 には，Buteo[10]というアドレス帳やカレンダーなどの情報を同期用フレームワークが既にインストールされている．それを利用することで，作業量を減らすことができる．

・ 独自の方法で実装

作業量は多くなるが，既存フレームワークの勉強は必要なくなる．また，依存性がないため，実装時のトラブルは少ないと想定される．

MeeGo コミュニティの公開情報[11]を調べた結果，MeeGo プラットフォームのバージョン 1.2 から，利用される同期フレームワークが変更されることが判明した.

MeeGo コミュニティは，Buteo を廃止し，Buteo に代わって SyncEvolution[12]というフレームワークへの切替作業を行っている．

以上の理由で，我々が作ったサービスが長期利用でき，MeeGo 自体の変更からも影響を受けないように，同期機能の設計は Buteo や SyncEvolution を参考にしつつ，独自で実装する方針を提案した．

● 通信フレームワークにおける調査提案

同期処理の実現方針より，Buteo と SyncEvolution を参考し，同期方式の設計・実装はできるが，サーバーとクライアントとの通信が課題となった．

まず，サーバーとクライアントとの通信データ形式の選定について述べる．

本システムの同期要求より，通信データは「命令」と「データ」の二つの種類を洗い出した．命令は「接続」,「ログイン」,「パスワード変更」などのもので,「データ」は命令の内容である．例えば,「ログイン」命令に対して，データは「ユーザ名」と「パスワード」となる．

また,「命令」ごとの「データ」の内容はそれぞれ異なる．また，将来的な機能追加により変更や追加などが良くあると想定する．

さらに，MeeGo デバイス以外のプラットフォームでも利用できるように，サーバーとクライアントとの結合度を最低限の情報的強度（特定のデータ構造）[13]に決定した．

以上の観点から，開発の柔軟性と拡張性を考慮し，通信データの形式は標準化された汎用的なデータ構成方法である XML 方式を決定した．

また，クライアントとサーバーの開発環境（.NetFramework,Qt）は XML データを操作するライブラリが充実しているため，開発の負担も減少する．

著者は XML をベースにしたオープンソースである通信フレームワークについて，調査を行い，本システムへの導入について評価を行った．

1. XMPP[14](Extensible Messaging and Presence Protocol)
2. SOAP[15](Simple Object Access Protocol)
3. Apache MINA[16]

以上の２と３は本システムのクライアントソフトの開発環境である QT 向けの事例が見つからなかったため，実装のリスクは結構高いと判別した．

また，１の XMPP は，QT 向けのオープンソースフレームワーク QXMPP[17]はあるが，まだ RC1（候補版）である，それに加えて XMPP 自体の勉強時間や MeeGo への導入条件などが不明であるため，実装のリスクも高いと判別した．

一方，本システムの XML 構造はあまり複雑ではなく，通信部分はフレームワークを利用せず，独自で実装することを一つの方法として提案した．

チーム内の検討した結果，通信データは XML ストリームで，通信自体は，フレームワークの代わりに，独自でソケット通信を実装するようにした．

ソケット通信を実装する方式では，フレームワークを利用する方式と比べると，通信の管理は独自で実装するため，多少手間がかかるが，フレームワークの勉強や改造などの時間は不要である．また，クライアント側の開発も QT フレームワーク基本の API を用いて実装が可能であるため，MeeGo デバイスへの導入のリスクは抑えられる．以上のことから，本システムに適した手法と考えられる．

また，クライアントとサーバーは特定の TCP ポートでソケット通信を行うため，社内ネットワークで本サービスを利用する場合，ファイアーウォール側で本システムが利用されるポートのアクセス権限の設定が必要となる．だが，本システムのクライアント対象デバイスは個人が所有するデバイスであるため，普段は家庭や公共ネットワークなど，TCP ポート制限がない環境で利用するため，本システムへの影響は微小と想定する．

6.2 暗号化機能の設計及び開発

本節では，著者が設計したデータ通信の暗号化方式について述べる．

6.2.1 暗号化における検討事項

著者は本システムの通信の暗号化方式を設計する時に，サーバーソフトウェアとクライアントソフトウェアそれぞれ，SSL[18]方式導入の実現性について調査実験を行った．

- クライアント側の開発環境 QT では QSslSocket クラスを用いて実装可能．
- サーバー側の開発環境.NetFramework では SSLStresm クラスを用いて実装可能．

著者は QSslSocket と SSLStresm を用いて事例プログラムを作成し，テストを行った．結果としては，QT 環境の QT サーバーと QT クライアント，.NetFramework 環境の.NetFramework サーバーと.NetFramework クライアントはそれぞれ通信できるが，QT と.NetFramework との通信はうまくできなかった．

その後，サーバーとクライアント両方利用できるオープンソースフレームワークである OpenSsl[19]における調査実験を行った．結果として，OpenSsl が提供した API を用いて，クライアントとサーバーとの暗号化通信は成功した．

そのため，本システムのサーバーとクライアントとのデータ通信の暗号化機能の実現方法は，標準 SSL の仕組みを参照して設計し，OpenSsl が提供された各種 API を利用して実装するように決定した．

6.2.2 暗号化機能の設計

まず，本システムが利用する暗号化方式について述べる．

- AES（Advanced Encryption Standard）方式[20]

  2000 年から規格化された，データの暗号化と復号化は同じ鍵で行う，いわゆる「対称」暗号方式である．1977 年に発行された同様の対称暗号方式である DES を進化した方式である．

  本システムはクライアントとサーバーとのデータ通信時は AES 方式を選定した．
- RSA 方式[21]

  1977 年に発明され，発明者である Ron Rivest，Adi Shamir と Len Adleman の頭文字をつなげて RSA と呼ばれる「非対称」暗号方式である．対称方式とは異なり，暗号と復号はそれぞれの鍵で行う方式である．例えば，公開鍵で暗号化した場合，復号は秘密鍵で行う．

本システムサーバー認証時には RSA 方式を選定した.

設計した暗号化処理の流れを図 6-2 に示す.

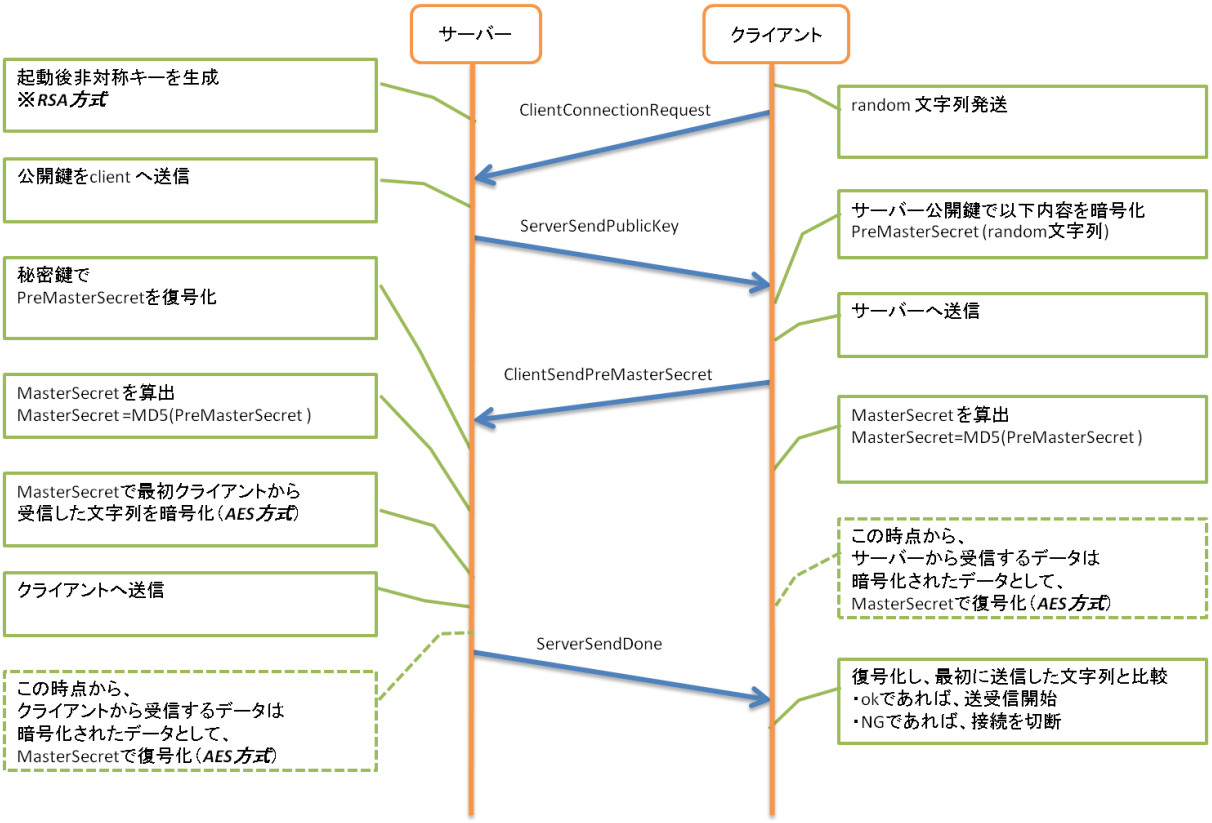


図 6-2　暗号化処理の流れ

クライアントは初期接続の時に，動的生成された文字列をサーバーへ送信する．その後，サーバーは公開鍵をクライアントへ送信し，クライアントはサーバーの公開鍵で通信用鍵（AES 鍵）を算出するためのデータを暗号化してサーバーへ送信する．

サーバーは秘密鍵でクライアントからのデータを復号し，通信用鍵（AES 鍵）を算出する．その後，通信用鍵（AES 鍵）を用いて最初にクライアントからの文字列を暗号化してクライアントへ送信する．また，サーバーソフトウェアは以降のクライアントとのすべての通信データを暗号化する．

また，クライアントは独自で算出した通信用鍵（AES 鍵）でサーバーからのデータを復号化する．前に発送した文字列と同様であれば，サーバー認証を成功として，以降サーバーとのすべての通信データを暗号化する．

以上の方式では，SSL の標準流れより省略した部分はいくつかあるが，サーバーとクライアントの実現はより容易であり，また，通信データは暗号化されるため，有効な方式であると考えられる．

6.3　クライアントテストプログラムの設計及び開発

本システムの開発フェーズでは，サーバーソフトウェアとクライアントソフトウェアの実装を並行的に行った．また，サーバーソフトウェアの開発はクライアントソフトウェアの開発より順調であったため，通信用 XML データの内部設計は著者が中心的となり，先行的に実装を行った．

サーバーソフトウェアのテストとクライアントソフトウェアの開発にサンプルを提供するため，著者がサーバーソフトウェアの開発と並行的にクライアントテストプログラムの開発を行った．

本節では，著者が設計開発したクライアントテストプログラムについて述べる．

- 画面設計

テストプログラムの画面を図 6-3 に示す．

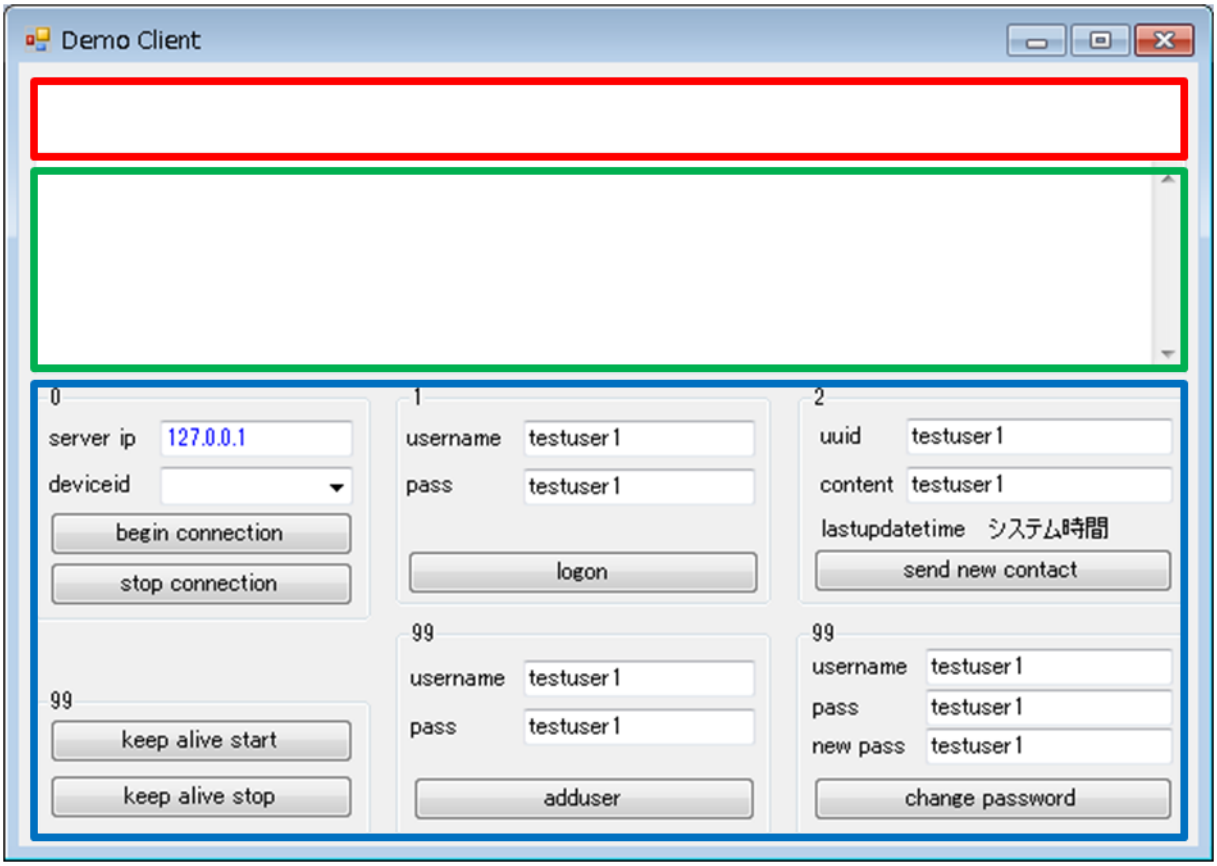


図 6-3　クライアントテストプログラム画面

画面構成は以下の三つの部分に分かれている．

・ 通信データの表示部分

クライアントからサーバーへ送信したデータ，サーバーから受信したデータや異常のメッセージなどを表示する部分である．より見やすくするため，表示内容の長さと関係なく，すべて改行せずに，一行で表示するようにした．

・ 受信データの表示部分

サーバーから受信したデータを重点的確認するための表示部分である．また，受信データは XML 形式であるため，表示する時には，XML 項目ごとに改行を行う．

・ 各種命令の実行部分

サーバーとクライアントとの通信命令とデータをサーバーへ送信する部分である．図に示したように，接続要求の送信とソケットの終了，ログイン，テストデータの送信，ユーザの登録, パスワードの変更や KeepAlive パケットの送信などの命令は実行できる．

また，各個命令のデータ部について，テキストボックスで手動入力も可能である．

- 機能設計

本テストプログラムのテスト範囲は，本システムのセキュリティ部分，XML データの送受信部分とソケット通信部分である．

本プログラムの「begin connection」ボタンを押下した場合，図 6-2 に示したクライアント側の暗号化処理を行う．

画面上の「logon」，「send new contact」，「adduser」などの命令ボタンを押下した場合，命令のXMLデータを生成し，バイト型の配列に転換，サーバーへ送信する．また，受信したバイト型のデータはXML文字列に転換してから画面に表示する．

ソケット通信はサーバーソフトウェアと同様に，4.3.1項で述べた非同期ソケット通信方式を実装し，サーバーソフトウェアの送受信をテストする．

6.4　データベースアクセスモジュールの基本設計

著者は基本設計フェーズで，本システムのデータベースを操作するアクセスモジュールのインタフェース設計を行った．アクセスモジュールはサーバーソフトウェアとデータベースとのやり取りを行うモジュールである．

設計したデータベースアクセスモジュールのクラス概要を図 6-4 に示す．

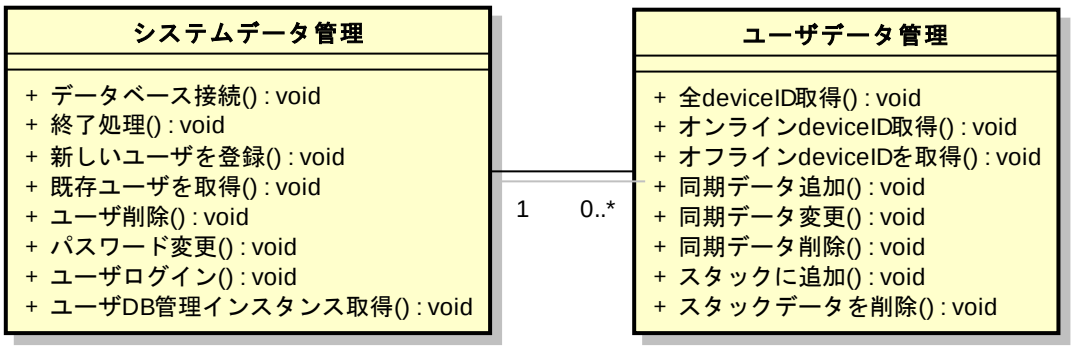


図 6-4　データベースアクセスモジュール概要クラス図

設計の目標は，サーバーソフトウェアで使いやすく且つ機能的な拡張性の高い設計であった．そのため，データベースへの操作処理を細かい粒度で，処理単位で洗い出した．また，目標は API を提供することであるため，簡潔なクラス構造に設計した．

システム DB 管理クラスは，ユーザ情報を集中的に管理するスキーマであるシステムスキーマにアクセスするための専用クラスである．

ユーザ DB 管理クラスは，ユーザデータを格納するユーザスキーマにアクセスするための専用クラスである．

サーバーソフトウェアは起動した直後，システムデータ管理クラスの唯一のインスタンスを作成する．

また，ユーザがログインした時に，システムデータ管理クラスのインスタンスは，ユーザデータ管理クラスのインスタンスを作成する．その後，サーバーソフトウェアはユーザデータ管理クラスから提供される API でユーザデータの操作を行う．

第7章　サーバーソフトウェアの性能評価

本章では著者が設計開発したサーバーソフトウェアの性能評価における評価環境，評価方法と評価結果について述べる．

7.1　評価環境

本評価を行う時のシステム環境については以下の通りである．

- サーバー環境

本システムのサーバーソフトウェアを実行するテストサーバーマシンの仕様を表 7.1 に示す．

表 7.1　テストサーバーの仕様

OS	Windows Server 2008/R2 /64Bit
メーカー	DELL
CPU	インテル® Core™2 Duo プロセッサーE7500（2.93GHz ）
メモリ	8GB

- クライアント環境

クライアント側はテスト用プログラムを用いて評価を行った．テスト用クライアントソフトウェアを実行するマシンは３台である．クライアントテストマシンの仕様を表 7.2 に示す．

表 7.2　クライアントテストマッシンの仕様

マッシン１	OS	Windows 7 Home /64Bit
	メーカー	Apple
	CPU	インテル® Core™2 Duo プロセッサーU9600（1.60GHz）
	メモリ	4GB
マッシン２	OS	Windows 7 Professional /32Bit
	メーカー	Panasonic
	CPU	インテル® Core™2 Duo プロセッサーU9400（1.40GHz）
	メモリ	4GB
マッシン３	OS	Windows 7 Professional /32Bit
	メーカー	Lenovo(IBM)
	CPU	インテル® Core™2 Duo プロセッサーT7200（2.0GHz）
	メモリ	2GB

- ネットワーク環境

本システムのサーバーは筑波大学構内の研究室にある．

クライアントマッシンは筑波大学学内にアクセスするため専用のソフトウェアである

PacketiX VPN を使ってサーバーと接続を行う.

7.2 評価項目

サーバーソフトウェアは同時に処理できるクライアント数及び多くのクライアントが同時に接続する時の処理時間の二つを検証した.

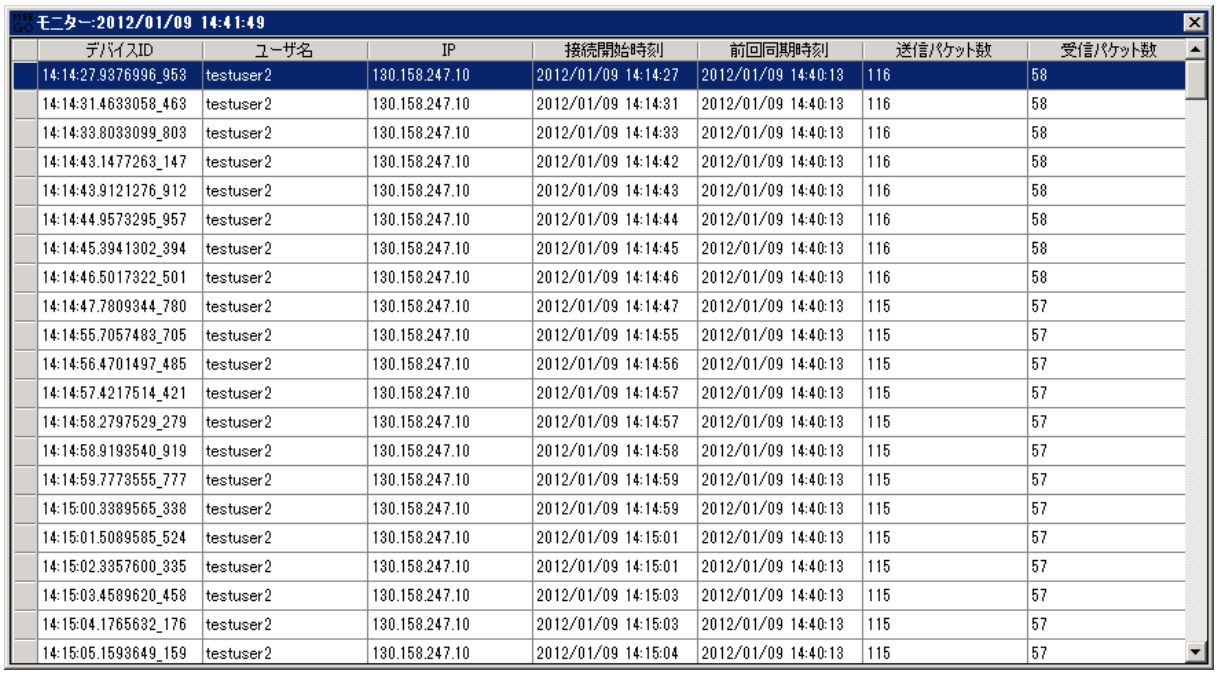
まずサーバーソフトウェアは同時に処理できるクライアント数について，テストクライアントソフトウェアを三台のクライアントテストマシンで各 100 個ずつ実行し, 同時接続した.

また，実行時間についての評価項目を以下に示す.

- クライアント接続数が 100 の場合，1 台クライアントは，1 件データ，10 件データ，100 件データをサーバーへ送信，処理結果を受信するまでの処理時間をそれぞれ 100 回測った.
- クライアント接続数が 100 の場合，10 台同時に 1 件データ，10 件データ，100 件データをサーバーへ送信，処理結果を受信するまでの処理時間をそれぞれ 10 回測った.
- クライアント接続数が 100 の場合，100 台同時に 1 件データ，10 件データ，100 件データをサーバーへ送信，処理結果を受信するまでの処理時間をそれぞれ 1 回測った.

7.3 評価結果

300 個クライアント同時に接続した際のモニタリング画面を図 7-1 に示す.


モニター:2012/01/09 14:41:49

デバイスID	ユーザ名	IP	接続開始時刻	前回同期時刻	送信パケット数	受信パケット数
14:14:27.9376996_953	testuser2	130.158.247.10	2012/01/09 14:14:27	2012/01/09 14:40:13	116	58
14:14:31.4633058_463	testuser2	130.158.247.10	2012/01/09 14:14:31	2012/01/09 14:40:13	116	58
14:14:33.8033099_803	testuser2	130.158.247.10	2012/01/09 14:14:33	2012/01/09 14:40:13	116	58
14:14:43.1477263_147	testuser2	130.158.247.10	2012/01/09 14:14:42	2012/01/09 14:40:13	116	58
14:14:43.9121276_912	testuser2	130.158.247.10	2012/01/09 14:14:43	2012/01/09 14:40:13	116	58
14:14:44.9573295_957	testuser2	130.158.247.10	2012/01/09 14:14:44	2012/01/09 14:40:13	116	58
14:14:45.3941302_394	testuser2	130.158.247.10	2012/01/09 14:14:45	2012/01/09 14:40:13	116	58
14:14:46.5017322_501	testuser2	130.158.247.10	2012/01/09 14:14:46	2012/01/09 14:40:13	116	58
14:14:47.7809344_780	testuser2	130.158.247.10	2012/01/09 14:14:47	2012/01/09 14:40:13	115	57
14:14:55.7057483_705	testuser2	130.158.247.10	2012/01/09 14:14:55	2012/01/09 14:40:13	115	57
14:14:56.4701497_485	testuser2	130.158.247.10	2012/01/09 14:14:56	2012/01/09 14:40:13	115	57
14:14:57.4217514_421	testuser2	130.158.247.10	2012/01/09 14:14:57	2012/01/09 14:40:13	115	57
14:14:58.2797529_279	testuser2	130.158.247.10	2012/01/09 14:14:57	2012/01/09 14:40:13	115	57
14:14:58.9193540_919	testuser2	130.158.247.10	2012/01/09 14:14:58	2012/01/09 14:40:13	115	57
14:14:59.7773555_777	testuser2	130.158.247.10	2012/01/09 14:14:59	2012/01/09 14:40:13	115	57
14:15:00.3389565_338	testuser2	130.158.247.10	2012/01/09 14:14:59	2012/01/09 14:40:13	115	57
14:15:01.5089585_524	testuser2	130.158.247.10	2012/01/09 14:15:01	2012/01/09 14:40:13	115	57
14:15:02.3357600_335	testuser2	130.158.247.10	2012/01/09 14:15:01	2012/01/09 14:40:13	115	57
14:15:03.4589620_458	testuser2	130.158.247.10	2012/01/09 14:15:03	2012/01/09 14:40:13	115	57
14:15:04.1765632_176	testuser2	130.158.247.10	2012/01/09 14:15:03	2012/01/09 14:40:13	115	57
14:15:05.1593649_159	testuser2	130.158.247.10	2012/01/09 14:15:04	2012/01/09 14:40:13	115	57

図 7-1 モニタリング画面

また，サーバーソフトウェアのメモリー利用量は 369728K(タスクマネージャーより)で，サーバーマシン全体の CPU 使用率は図 7-2 の通りである.

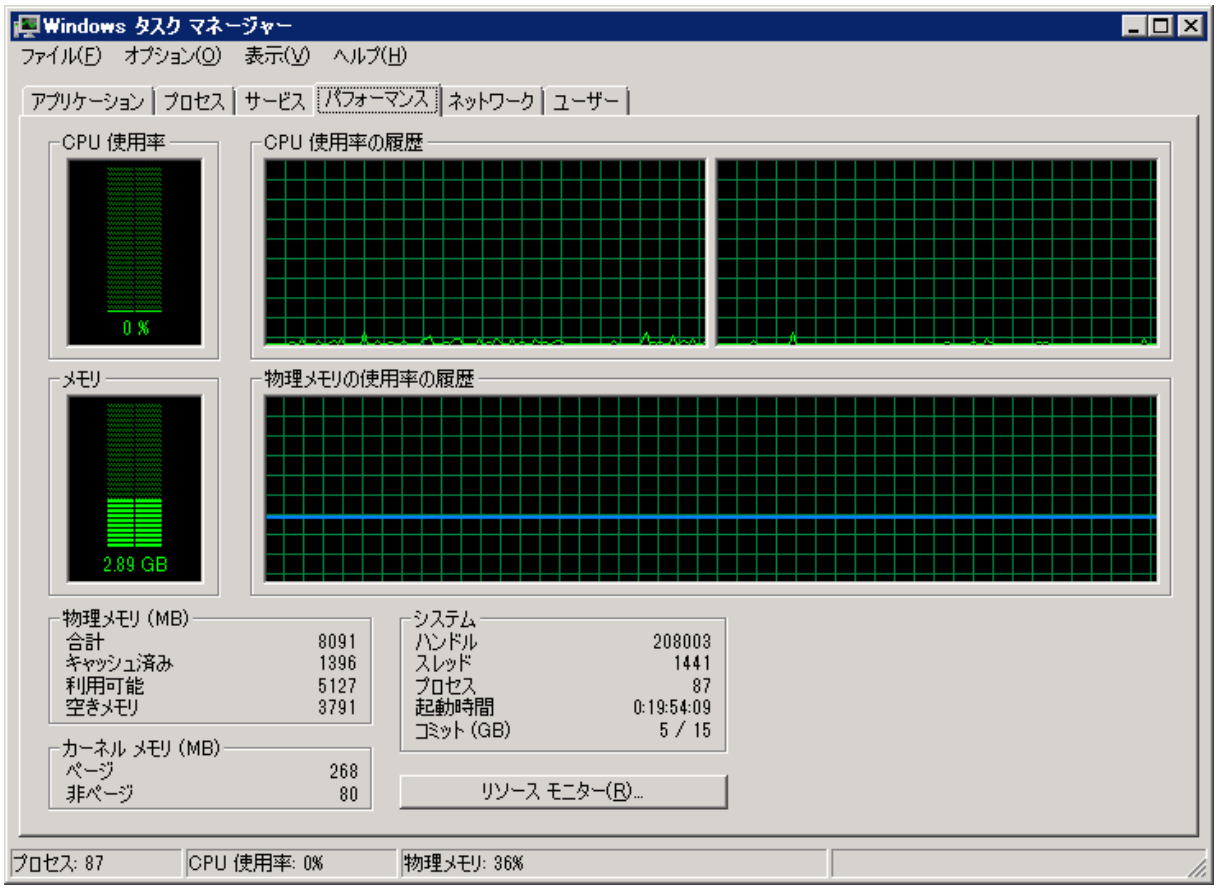


図 7-2　CPU 使用率

また，各種処理時間テストの最大，最小，平均処理時間と処理全体の中の I/O 処理（データベース操作）時間の比率を表 7.3 に示す.

その内，同時送信クライアント数は 100 の場合，テストマシン三台ともにデータ処理中では CPU 使用率が 100%であった．そのため，同時送信クライアント数は 100 の場合に測ったデータは予想より大きいであった．特に 100 件データを送信する場合，最大処理時間（572.790 秒）と平均処理時間（289.483 秒）は予想より結構大きかった.

表 7.3　実行時間の評価結果

接続クライアント数	同時送信クライアント数	一回送信データ数	最大処理時間(秒)	最小処理時間(秒)	平均処理時間(秒)	I/O 処理時間比率
100	1	1	0.173	0.021	0.032	>70%
		10	0.770	0.256	0.363	
		100	4.091	2.551	3.053	
100	10	1	0.564	0.031	0.188	
		10	3.962	0.265	1.823	
		100	30.478	0.671	15.063	
100	100	1	3.166	2.511	2.696	
		10	28.540	2.584	15.149	
		100	572.790	3.436	289.483	

サーバーソフトウェアが 1 件データを処理する場合，処理時間の分布を図 7-3 に示す.

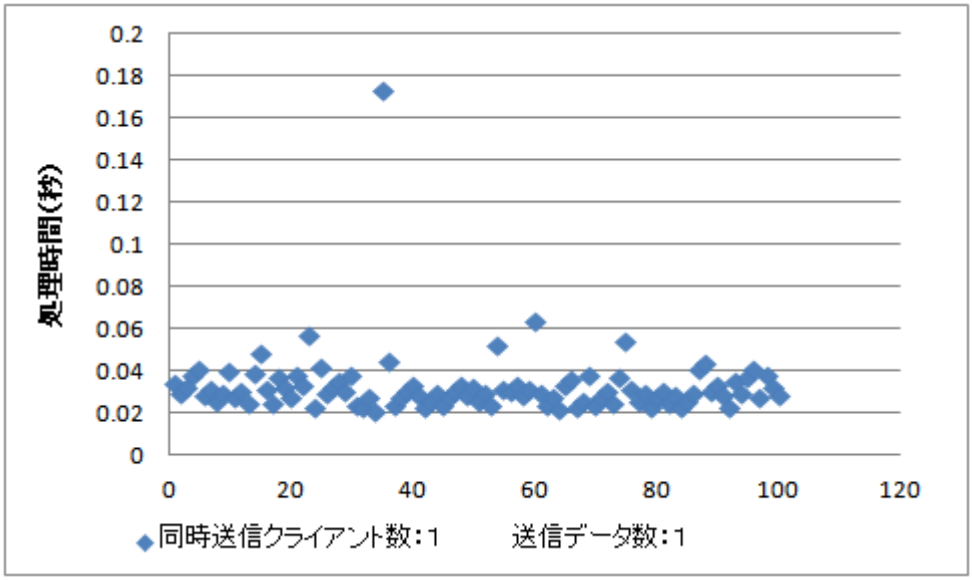


図 7-3　1 件データを処理する場合

データは 0.02 秒から 0.04 秒までの範囲に収まった.

サーバーソフトウェアは 10 件データを同時に処理する場合，処理時間の分布を図 7-4 に示す.

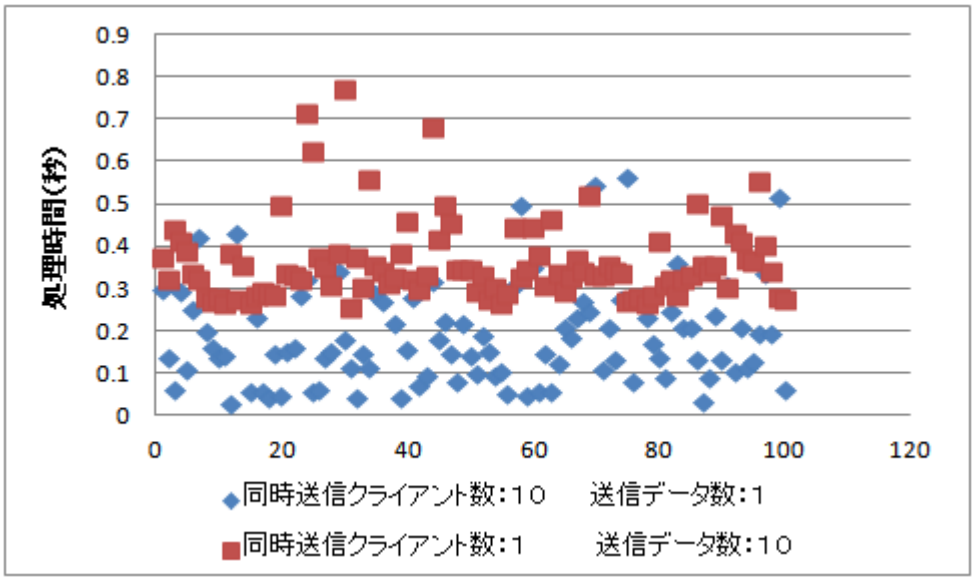


図 7-4　10 件データを処理する場合

同時送信クライアント数が少ない方がばらつきは小さい.

サーバーソフトウェアが 100 件データを同時に処理する場合，処理時間の分布を図 7-5 に示す.

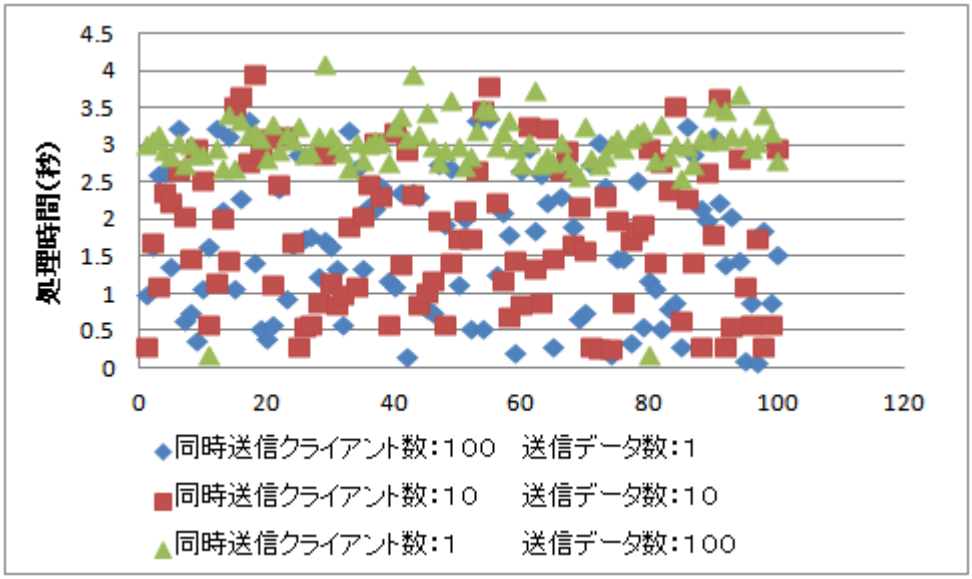


図 7-5　100 件データを処理する場合

図 7-4 と同じように，同時送信クライアント数が少ない方がばらつきは少ない.

サーバーソフトウェアが 1000 件データを同時に処理する場合，処理時間の分布を図 7-6 に示す.

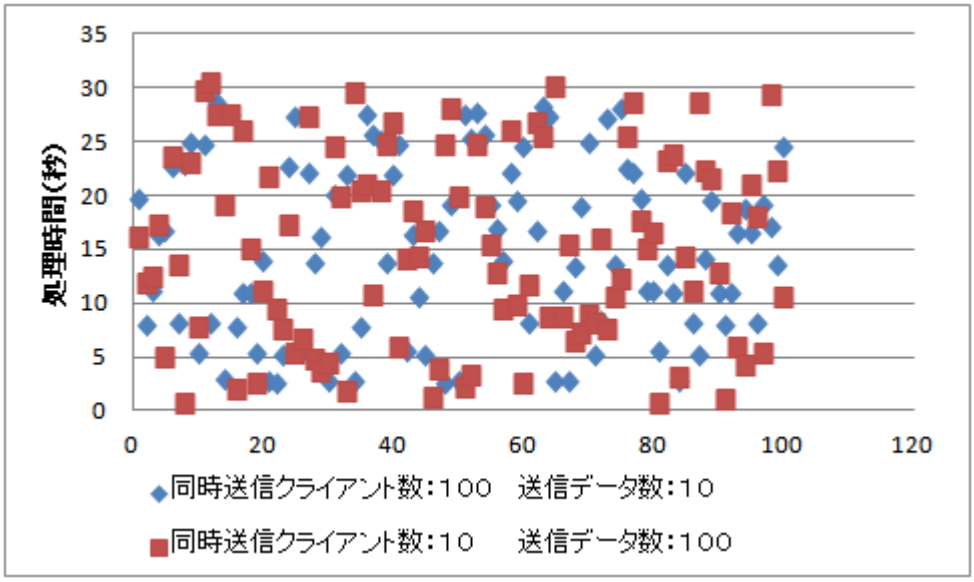


図 7-6　1000 件データを処理する場合

二つのパターンともにばらつきである.

サーバーソフトウェアが 10000 件データを同時に処理する場合，処理時間の分布を図 7-7 に示す.

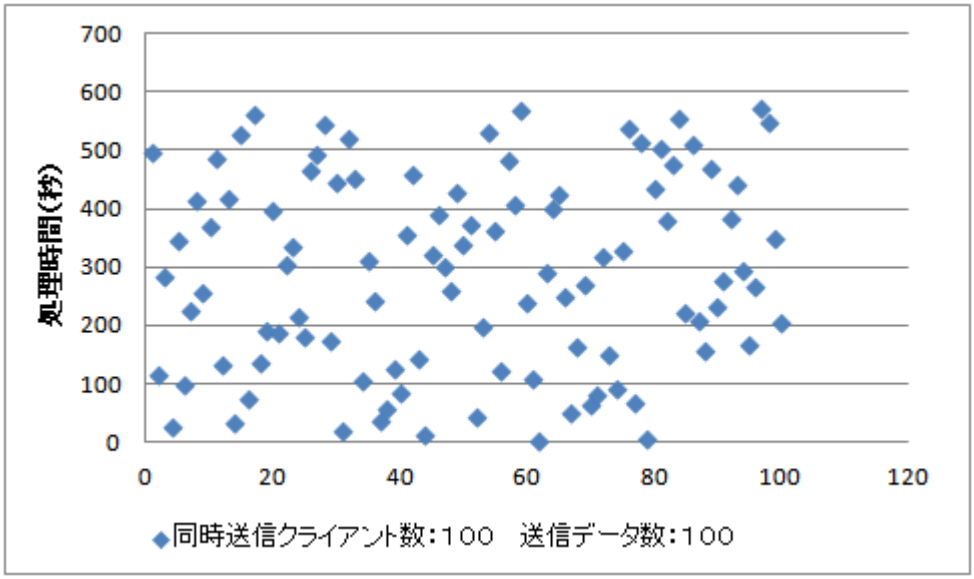


図 7-7　10000 件データを処理する場合

7.4　考察

実験結果により，同時に 100 件のデータを処理すると，最短は 0.265 秒で，最長は 4.091 秒である．本システム設計時の目標に満たしたと考えられる.

また，サーバーソフトウェアの性能に大きく影響した点を以下に示す.

・ I/O 処理時間

すべての平均処理時間の中の I/O 処理時間（データベースの処理時間）は予想以上に高い比率であった.

- クライアントデータの処理ロジック
  本サーバーソフトウェアを設計する時に，セッションごとの個別スレッド方式ではなく，すべてのセッションのデータ処理は同じスレッドである共有スレッド方式を選定した．
  そのため，サーバーソフトウェアの CPU 使用率は減る一方，データ処理は順序であるため，ばらつきの結果は多かった．

今後の課題として，改善できる点を以下に示す．

- I/O 処理を減る
  DB アクセスライブラリの開発担当者と相談し，処理ロジックの改造を行う．
- データの処理ロジックを変える
  すべてのセッションの共有スレッド方式より，ある程度の並列処理を行えば，処理時間を減らすことができると考えられる．

第8章　　プロジェクトの振り返り

本章では，サーバーソフトウェア開発の計画と実績，チーム作業や各個フェーズごとの振り返りを述べる．

8.1　　サーバーソフトウェア開発の計画と実績

本節では著者が担当したサーバーソフトウェア開発の計画と実績について述べる．

8.1.1　　開発スケジュールと実績

本システムのサーバーソフトウェアの開発スケジュールを図 8-1 に示す．

図 8-1　　開発スケジュールの計画と実績

計画立案からクラス設計までは 8 月初旬までの計画だったが，立案と案の選定は時間がかかったため，遅れが発生した．しかし，技術調査と開発，テストが順調であったため，予定より早めに完了した．

8.1.2 成果物

● ドキュメント資料

サーバーソフトウェアを開発中に以下のドキュメントを作成した.

- 要件定義フェーズでは，要件定義書，ユースケース図とユースケース記述をチーム共同で作成した.
- 基本設計フェーズでは，基本設計書，アクティブティ図と概要クラス図及び説明をチーム共同で作成した.
- UI 設計フェーズでは，サーバーソフトウェアの画面デモ及び定義書を作成した.
- 詳細設計フェーズでは，詳細設計書，詳細クラス図，インスタンス図，シーケンス図を作成した.
- テストフェーズでは，単体テスト仕様書と結合テスト仕様書を作成した.
- 評価フェーズでは，サーバーソフトウェア評価表を作成した.

● ソースコード

サーバーソフトウェアの実装フェーズでのモジュールごとの実績を表 8.1 に示す.

表 8.1 サーバーソフトウェア実装実績

モジュール	ステップ数
XML 処理	1798 ステップ
接続管理	317 ステップ
通信処理	476 ステップ
暗号化	174 ステップ
画面表示	126 ステップ
その他	247 ステップ

8.2 チーム作業の振り返り

● 良かった点

開発における技術ドキュメント，サンプルソースコードなど個人各々の一時的使用する資料を含めて，すべての資料はファイル共有ツールを利用して，チームメンバーに公開してきた．そうすることで，チーム内で知識の共有ができた.

また，例会時間以外で，メンバー同士が離れていて集まれない場合に急に出た技術の問題を解決する際は，音声通話ソフト Skype と vpn/vnc などのツールを利用して，共有デバッグ

環境を構築することで，共同作業を行った．

- 改善できる点

開発後半では，各自大きい粒度で進捗を報告したので，プロジェクト全体のスケジュール管理は不十分であった．原因としては，各自が分担された部分を中心的に作業したためである．開発ペースはばらばらであるからこそ，小さい粒度のスケジュールを立てて，定量的に作業を行えば，効率はもっと向上できると考えられる．

8.3 技術調査の振り返り

- 良かった点

MeeGo コミュニティに流れた公開メールを把握することで，MeeGo プラットフォームで利用する同期フレームワークについての切換えについての情報を確認できた．その後，開発フェーズに入る前に，同期処理は既存のフレームワークを利用せず，独自の方法で実現することを提案し，手戻りのリスクを抑えた．

- 改善できる点

スケジュールを立てる時に，開発や実装，評価を長めにするため，通信フレームワークの技術調査期間を２週間とたが，2 週間で複数フレームワークを評価した結果，利用できそうなものは見つからなかった．結果として独自の実装は多くなるため，実装とテストの工数は増えた．

通信フレームワークは多様なものがあるため, もっと時間を取ってじっくり調査を行えば, 本システムに適用できるものもあるかもしれない．そうすれば，実装工数は減られると考えられる．

8.4 開発の振り返り

- 良かった点

開発言語は著者が得意な言語 C#を採用したため，早いペースで初版のサーバーソフトウェアを実装できた．また，作成したクライアント用テストプログラムは，サーバーソフトウェアのテスト用以外のクライアントソフトウェアの開発メンバーにも参考になった．

セキュリティの実装においては，サーバー開発環境の.NetFramework とクライアント開発環境の QT，両方の暗号化・復号化をデバッグモードで実験しながら，開発を行った．クライアント開発のメンバーは著者が作成したソースコードで順調に実装できた．

- 改善できる点

クライアントソフトウェアの開発は, 予定通り実装できなかった技術問題が多く発生した．まだ MeeGo 向けソフトウェア開発に馴染めてないのも原因の一つであるが，予想以上の時間をかけても問題は解決できない場合は，はやめに MeeGo コミュニティやインテルの方に聞くべきと思われる．やれる範囲を先にどんどん進めて，より効率良く作業をするべきだったと考えられる．

8.5 評価の振り返り

- 良かった点

サーバーソフトウェアの新バージョンとデータベースアクセスモジュールの新バージョンとの結合は一発で，スムーズに統合できた．これは，お互いにきちんと設計した API 仕様書

に従って実装したことで，より理想なチームワークであるかと考えられる．

また，サーバーソフトウェアの評価フェーズでは，テスト用クライアントソフトウェアを用いて，スムーズに進行できた．

- 改善できる点

サーバーソフトウェアのテストについて，利用ユーザ数と実機数の制限があるため，著者は作成したクライアントテストプログラムを用いてテストを行ったが，多くの実機でテストを実施すべきであった．また，実験内容について，異なる種類の操作方法を多く実施すると，より現実に近い運用環境であるかと考えられる．

第9章　結言

本プロジェクトでは，インテルが提示した「オープンソフトウエアプラットフォームである MeeGo をベースにした次世代組み込みデバイス向けユーザ体験開発への挑戦」と言うテーマの元，MeeGo デバイス向けソフトウェアの計画立案を行った．

計画した案の中から個人データを自動共有する「データ共有システム」を選定し，設計開発を行った．

開発において，著者はサーバーソフトウェアの設計・実装，サーバーとクライアントとの通信方式の選定，セキュリティの設計・実装を行った．

開発したプログラムの信頼性について，テストプログラムを用いて評価を行った．その結果，より多くのクライアント同時接続時でも正しく処理すること，処理時間は予定通りであることが確認できた．また，通信データの形式はより高い汎用性である XML 形式で実装したため，MeeGo クライアント以外のほかの種類のクライアントでも対応できる．さらに，通信処理と XML 処理やセキュリティ部分では，他の用途でも参考できる．

サーバー側の開発はほぼ順調に遂行できた一方，クライアントの技術調査は予想以上，数倍時間がかかった．途中で調査を中止し，他の方法で実現する部分もあった．その分は今後の課題として，委託元のインテル社へ報告して，MeeGo の次世代プラットフォームである TiZen では改善して頂くことを期待する．

本プロジェクトの実施にあたりチーム作業の進めかた，オブジェクト指向設計，XML データの処理，ソケット通信や接続管理，通信セキュリティ，サーバーの構築など多くの知識と経験を得る事ができた．また，技術調査フェーズや開発中に色々調査を行って，その中で得た知識だけではなく，身に付けた自主的学ぶ方法，問題に対する分析・解決方法も今後に大変参考になる．

謝辞

尹栄植様，池井満様はじめとするインテル株式会社東京支社インテル技術本部ソフトウェア＆サービス技術統括部の皆様には，テーマの提供から，勉強会の開催，細部にわたる技術の支援，指導を頂き，深く感謝いたします．

指導教員の田中二郎教授には，二年間丁寧かつ熱心なご指導を賜りました．心から感謝申し上げます．

本プロジェクトの課題担当教員の追川修一先生にプロジェクトの熱心な指導や貴重な意見を賜りました．心から感謝申し上げます．

本報告書執筆にあたり，副査としてご助言を戴くとともに本報告書の細部にわたりご指導を戴いた中沢研也先生に深く感謝致します．

また，本プロジェクトのチームメンバーである刘建宇氏，許先華氏，森本悠矢氏から多くの力やご意見を頂き，大変お世話になりました．有難うございました．

最後に，さまざまな面で支えて頂いた彼女，家族，友人などすべての方々に心より感謝致します．

参考文献

[1] Linux Foundation, About, meego.com, https://www.meego.com/about

[2] インテル株式会社, 製品情報, http://www.intel.com/jp/products/processor/atom/

[3] ARM Holdings, プロセッサ, http://www.arm.com/ja/products/processors/index.php

[4] アップル株式会社, icloud-概要, apple.com, http://www.apple.com/jp/icloud/

[5] multisoft-lab, ソケットの使用法, http://www.multisoft-lab.com/voicechat/socket.html

[6]Blum, Richard, C# Network Programming, Sybex Inc, 2002.

[7] MSDN ライブラリ, Asynchronous Server Socket Example, microsoft.com, http://msdn.microsoft.com/ja-jp/library/fx6588te(v=VS.100).aspx

[8] Windows Dev Center-Desktop, Synchronous and Asynchronous I/O, microsoft.com, http://msdn.microsoft.com/en-us/library/windows/desktop/aa365683(v=vs.85).aspx

[9] Linux Foundation, meego-app-contacts, http://meego.gitorious.org/meego-ux/meego-app-contacts

[10] Buteo, meego.com, http://wiki.meego.com/Buteo

[11] MeeGo-dev, Some architecture changes, http://lists.meego.com/pipermail/meego-dev/2011-March/481890.html

[12] syncevolution.org, Development, http://syncevolution.org/development

[13]SCC 出版局, 改訂システム開発の基礎(情報処理基礎講座), 電子開発学園出版局出版, 2005.

[14] XMPP Standards Foundation, About, http://xmpp.org/about-xmpp/

[15] W3C recommendation, SOAP Version 1.2 Part 0: Primer (Second Edition), w3.org, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

[16] Apache Software Foundation, Features, http://mina.apache.org/features.html

[17] manjeetdahiya, jeremy.laine, ian.geiser, QXmpp, google.com, http://code.google.com/p/qxmpp/

[18]Viega, John　Messier, Matt　Chandra, Pravir, Network Security with Openssl, Oreilly & Associates Inc, 2002.

[19] OpenSSL Software Foundation, http://www.openssl.org/

[20] NIST, Federal InformationProcessing Standards Publication 197, Announcing theADVANCED ENCRYPTION STANDARD(AES), November 26, 2001.

[21] Tom Davis, RSA Encryption, http://www.geometer.org/mathcircles, October 10, 2003.

付録

開発ドキュメントリスト

バージョン	1.0
作成者	劉宏
レビュー者	許、森本、劉建
日付	2012/01/17

ID	フェーズ	ドキュメント名
01_001	要件定義	要件定義書
		ユースケース図
		ユースケース記述
02_001	基本設計	基本設計書
		アクティビティ図
		概要クラス図及び説明
02_002		IF 定義書(DB アクセスライブラリ)
03_001	UI 設計	画面デモ及び定義書(サーバー)
03_002		画面デモ及び定義書(クライント)
04_001	詳細設計	詳細設計書(サーバー)
04_002		詳細設計書(クライント)
04_003		暗号化
		詳細クラス図
		インスタンス図
		シーケンス図
04_004		DB モジュール仕様書
04_005		XML 仕様
05_001	DB 設計	DB 設計仕様書
		E・R 図
06_001	実装	コーディング規約(クライアント)
06_002		コーディング規約(サーバー)
07_001	単体テスト	単体テスト仕様書(サーバープログラム)
07_002		単体テスト仕様書(DB 操作モジュール)
07_003		単体テスト仕様書(クライアントコンタクト、通信と内部処理)
07_004		単体テスト仕様書(同期モジュール)
08_001	結合テスト	結合テスト サーバーソフトと DB
08_002		結合テスト クライントソフトと画面
08_003		結合テスト サーバーソフトとクライントソフト
08_004		バグ連絡表
09_001	評価	サーバー評価表
09_002		クライント評価表
99_001	その他	スケジュール管理表

【データ共有システム要件定義書】

文書番号

01-001

	作成
作成者名	森本悠矢
修正者名	劉　宏超 劉　建宇 許　先華
作成日	2011 年 8 月 10 日

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考
1	2012.1.13	v1.1	シーケンス図とシーケンス記述	

目　次

1　概要

1.1　目的

本ソフトウェアは MeeGo 端末上にあるアドレス帳、カレンダー、ノート、ウェブ履歴のコンテンツを、同一ユーザが持つ複数の MeeGo 端末上で共有することを目的としている。これにより、複数の端末間で煩雑な操作をすることなく、自然に作業を移行することが可能になる。

本ソフトウェアは MeeGo 端末上で動作するクライアントアプリケーションとサーバアプリケーションで構成される。

1.2　対象ユーザ

本ソフトウェアは MeeGo を搭載したタブレットを持つユーザを対象としている。

1.3　記載範囲

本ドキュメントではソフトウェアの要件定義について記述する。

1.4　参照ドキュメント

ID	文書名	文書番号	発行年月	備考

1.5　定義（用語、略語）

ID	用語・略号	正式表記	意味
	MeeGo	MeeGo	オープンソフトウェアプラットフォーム

2　機能概要

本ソフトウェアは複数の MeeGo 端末上でアドレス帳、カレンダー、ノート、Web の閲覧履歴を共有する機能を持つ。サーバと通信する際のユーザ ID、パスワードの設定、通信の開始・停止の操作は MeeGo のデスクトップに設定パネルを一つ追加し、そのパネル上で各操作を行う。

3　制約条件

3.1　ハードウェアなどを含めたシステム全体の構成と制約

本ソフトウェアのクライアントはネットワークに接続された MeeGo タブレットで動作する。

本ソフトウェアのサーバは Windows 上で動作する。

DB は MySQL のバージョン 5.5 を使用する

ネットワークを通じてアドレス帳などの個人情報がやり取りされるため、通信の際には暗号化などを行い高い信頼性を持つことが必要となる。

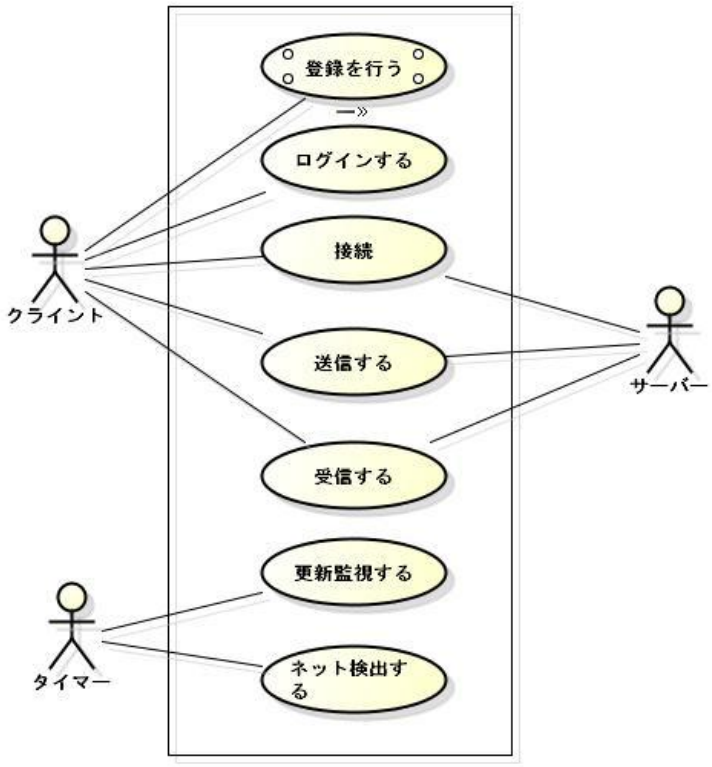
また、法的な制約として、第三者に著作権が存在するメディアファイルを複数のユーザがアクセス可能なサーバ上にアップロードすると、公衆送信権の侵害となる可能性が高い。よって、音楽、動画などのメディアファイルはサーバへのアップロードは行わない。

4　構成

4.1　周辺システム、利用するハードウェアおよびソフトウェア

種類	名称	バージョン
利用する OS（クライアント）	MeeGo Tablet	1.2.0.90
利用する OS（サーバ）	Windows Server 2008	R2 64Bit
利用するミドルウェア	MySQL	5.5
開発言語（サーバ）	C#	-
開発環境（サーバ）	Microsoft Visual Studio 2010	
開発言語（クライアント）	C++(Qt)	
開発環境（クライアント）	Ubuntu Desktop	10.04
	MeeGo SDK	1.2

5　機能要件詳細（ユースケース図とユースケース記述）



ユースケース ID	ユースケース名	概要
001	サーバ接続	サーバーとの接続を確立する
002	クライアント接続	クライアントとの接続を確立する
003	ログイン（クライアント）	サーバと接続を確立し、共有を開始する
004	ログイン（サーバー）	クライアントと接続し、共有を開始する
005	ネットワーク検出	自動的にネットワークに接続する
006	登録（クライアント）	ユーザの id、password をサーバーに登録する
007	登録（サーバー）	ユーザの id、password をデータベースに登録する
008	更新監視	サーバへアップロードすべきデータがないか定期的に確認する
009	サーバーへの送信	MeeGo 端末上のコンテンツをサーバにアップロードする
010	クライアントから受信	MeeGo 端末上のコンテンツをサー

		バにアップロードする
011	サーバーから受信	サーバ上のコンテンツを MeeGo 端末にダウンロードする
012	クライアントへ送信	サーバ上のコンテンツを MeeGo 端末にダウンロードする

ユースケース ID:001
ユースケース名:サーバ接続
目的: サーバとの接続を確立する
アクター:クライアントプログラム
事前条件:ID とパスワードが入力済み
基本系列
1. ID とパスワードを暗号化する
2. サーバに ID とパスワードを送信する
3. サーバから接続完了の通知が来るのを待つ
4. サーバとの通信を開始する

代替系列
2 - a サーバが見つからなかった場合
1. ユーザにその旨を通知し、ネットワークの確認をしてもらうようダイアログで通知する
2. 当該ユースケースを終了する

2 - b ユーザが見つからなかった場合
1. ID とパスワードを確認する旨をダイアログで通知する
2. 当該ユースケースを終了する

事後条件: サーバとの接続が確立されている

ユースケース ID:002
ユースケース名：クライアント接続
目的：クライアントとの接続を確立する
アクター：サーバプログラム
事前条件：ソフトウェアが起動している
基本系列
1. 利用者からアクセス要求が来るのを待つ
2. ID とパスワードを復号し、データベースでユーザを確認する
3. ユーザをログイン状態にする
4. クライアントに接続完了の旨を通知する
代替系列
2 - a ID とパスワードに該当するユーザが見つからなかった場合
    1. クライアントにログインができなかった旨を通知する
    2. 基本系列 1 に戻る
事後条件：クライアントとの接続が確立されている

ユースケース ID:003
ユースケース名：ログイン（クライアント）
目的：サーバと接続を確立し、共有を開始する
アクター：クライアントプログラム
事前条件：ソフトウェアが起動している
基本系列
5. 端末に保存された id、password を暗号化する
6. サーバに暗号化されたデータを送信する
7. サーバから接続完了の通知を待つ
8. 前回終了時のデータの更新時間と現在のデータの更新時間比較する
9. 更新されたデータをサーバにアップロードする
10. 更新監視を開始する
事後条件：サーバにログインが完了している

ユースケース ID:004
ユースケース名：ログイン（サーバー）
目的：クライアントと接続し、共有を開始する
アクター：サーバプログラム
事前条件：ソフトウェアが起動している
基本系列
1. クライアントからデータが送信されてくるのを待つ

2. 受信されたデータを復号する
3. ログイン用のデータであることを確認し、DB 上のログイン状態をオンにする
4. 接続完了通知をクライアントに送信する

代替系列

3 - a 接続ができなかった場合

3. 基本系列 2 に戻り、再度接続を行う

5 - a 更新すべきデータがなかった場合

1. 基本系列 6 に移行する

事後条件：クライアントと接続され、共有が開始されている

ユースケース ID:005
ユースケース名：ネットワーク検出
目的： 自動的にネットワークに接続する
アクター：クライアントプログラム
事前条件：ソフトウェアが起動しており、かつネットワークに接続されていない
基本系列
1.定期的にネットワークの接続状態を監視する
2.ネットワークに接続されていることが確認された場合、ログイン処理を実行する
事後条件： サーバにログインし、共有が開始されている

ユースケース ID:006
ユースケース名：登録（クライアント）
目的： ユーザの id、password をサーバに登録する
アクター：クライアントプログラム
事前条件：ソフトウェアが起動しておりサーバと接続されている
基本系列
1. ユーザがテキストボックスに id、password を入力する
2. サーバに入力された id、password を送信する
3. 登録確認通知を待つ
4. 登録が確認された場合、ログイン作業を行う

代替系列

4 - a 登録できなかった場合

1. ユーザに id が使用できない旨をダイアログで伝える
2. 基本系列 1 に戻り、再度 id、password の入力を促す

事後条件： サーバにユーザの id、password が登録されている

ユースケース ID:007
ユースケース名：ユーザ登録（サーバー）
目的：ユーザの id、password を登録する
アクター：サーバプログラム
事前条件：ソフトウェアが起動している
基本系列
5. クライアントからデータが送信されてくるのを待つ
6. 受信されたデータを復号する
7. 登録用のデータであることを確認した場合、id がそれまでに登録されていないかチェックする
8. id、password を DB に登録する
9. 登録確認通知をクライアントに送信する
代替系列
3 - a id が既に登録されていた物だった場合
    1. クライアントに登録不可能の通知を送信する
    2. 基本系列 1 に戻る
事後条件：サーバが新規ユーザのデータを登録済みである

ユースケース ID:008
ユースケース名：更新監視
目的：サーバへアップロードすべきデータがないか定期的に確認する
アクター：クライアントプログラム
事前条件：ソフトウェアが起動しておりサーバと接続されている、該当コンテンツが共有可能になっている
基本系列
5. 一定時間ごとにファイル・フォルダの更新日時を調べ、前回確認時の更新時間と比較する
6. 更新すべきデータをサーバへアップロードする
7. 確認用データの更新日時を更新する
事後条件：サーバ上に最新のファイルがアップロードされている

ユースケース ID:009
ユースケース名：サーバへ送信
目的：MeeGo 端末上のコンテンツをサーバにアップロードする
アクター：クライアントプログラム
事前条件：ソフトウェアが起動しておりサーバと接続されている、該当コンテンツが共有可能になっている

基本系列
8. XML に変換する
9. XML を暗号化する
10. サーバに XML をアップロードする

代替系列
1 - a サーバとの接続が切断された場合
3. それまでに受信したデータを削除する
4. サーバとの接続を試みる
5. 接続が成功した場合、基本系列 1 に戻り更新が必要なコンテンツを確認する

1 - b コンテンツが現在の状態で最新だった場合
3. サーバに更新が不必要であると通知する
4. 当該ユースケースを終了する

3 - a アップロードに失敗した場合
1. サーバとの接続を確認する
2. サーバと正しく接続されている場合、基本系列 3 に戻り再度コンテンツをアップロードする

事後条件: サーバに MeeGo 端末上のコンテンツが最新の状態でアップロードされている。

ユースケース ID:010
ユースケース名:クライアントから受信
目的: MeeGo 端末上のコンテンツをサーバにアップロードする
アクター:サーバプログラム
事前条件:ソフトウェアが起動しておりクライアントと接続されている、該当コンテンツが共有可能になっている
基本系列
10. クライアントから更新を行うコンテンツが送信されてくるのを待つ
11. 受信したデータを復号する
12. XML を解析し、更新するデータを取得する

代替系列
1 - a クライアントとの接続が切断された場合
3. 該当のクライアントの接続状況をオフにする
4. 基本系列 1 に戻る

1 - a コンテンツの受信に失敗した場合
1. クライアントとの接続を確認する
2. 基本系列 1 に戻る

事後条件：サーバが MeeGo 端末上のコンテンツを最新の状態で保存している

ユースケース ID:011
ユースケース名：サーバから受信
目的：サーバ上のコンテンツを MeeGo 端末にダウンロードする
アクター：クライアントプログラム
事前条件：ソフトウェアが起動しておりサーバと接続されている、該当コンテンツが共有可能になっている
基本系列
11. サーバからデータが送信されるのを待つ
12. 送信されてきたデータを復号する
13. XML を解析する
ファイルの場合
14. 一時ディレクトリに保存する
15. 一時ディレクトリからファイルディレクトリにコンテンツをコピーする
16. 一時ディレクトリの送信されてきたファイルを削除する
ファイル以外の場合
17. 解析したデータを追加・更新する
代替系列
1 - a 受信中に接続が切断された場合
6. それまでに受信したデータを削除する
7. サーバとの接続を試みる
8. 接続が成功した場合、基本系列 1 に戻る
5 - b コンテンツのコピーに失敗した場合
5. ファイル名の末尾に更新日時を付与し、再度コピーを試みる
6. コピーに失敗した場合、基本系列 1 に戻る
7. コピーに成功した場合、基本系列 6 に戻る
事後条件：MeeGo 端末上に最新のデータが保存されている

ユースケース ID:012
ユースケース名：クライアントへ送信
目的：サーバ上のコンテンツを MeeGo 端末にダウンロードする
アクター：サーバプログラム
事前条件：ソフトウェアが起動しておりクライアントと接続されている、該当コンテンツが共有可能になっている、新たに更新されたコンテンツが存在する
基本系列
13. コンテンツを XML 化する

14. コンテンツを暗号化する
15. 接続されているクライアントにコンテンツを送信する

事後条件： MeeGo 端末上に最新のデータが保存されている。

6　性能・品質等非機能要求詳細

6.1　信頼性に関する要求

各ファイル・履歴情報が最新のものになるように留意する必要がある。また、途中で通信が切断された場合でも、各ファイルを破壊せず通信が終わったファイルだけ更新できるようにしなければならない。

6.2　効率性に関する要求

クライアント側の他の処理を阻害しないように処理を行う必要がある。

また、容量の大きいファイルは分割して送信し、ユーザの通信を阻害しないようにする必要がある。

サーバ側の同時接続クライアント数は 100 台とする。また、同時処理データ数は 100 件とする。

6.3　保守性・移植性に関する要求

クライアントソフトウェアはタブレット以外の端末へも容易に移植可能にする必要がある。

6.4　セキュリティ面に関する要求

アドレス帳など個人情報をやり取りするため、通信時は暗号化を行い盗聴されないように留意する必要がある。

サーバ上の別のユーザの情報に誤ってアクセスしないよう、通信時にはユーザ認証を行う必要がある。また、id とパスワードとデータ内容は暗号化して保管し、管理者側からも見えないようにする。

データは全て DB に保存し、ユーザごとに DB を切り分けておくことでユーザのデータに他のユーザからアクセスできないようにする。

文書名

【データ共有システム基本設計書】

文書番号

02-001

	作成
作成者名	森本悠矢 劉　宏超 劉　建宇 許　先華
作成日	2011 年 9 月 1 日

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考

目　次

1　概要

1.1　目的

本ソフトウェアは MeeGo 端末上にあるドキュメント、アドレス帳、カレンダー、メディアファイル・Web の閲覧履歴などのコンテンツを、同一ユーザが持つ複数の MeeGo 端末上で共有することを目的としている。これにより、複数の端末間で煩雑な操作をすることなく、自然に作業を移行することが可能になる。

本ソフトウェアは MeeGo 端末上で動作するクライアントアプリケーションとサーバアプリケーションで構成される。クライアントプログラムは MeeGo のパネルに共有設定を行うためのパネルを追加し、そこで設定された情報を元にバックグラウンドでサーバと通信を行う。

1.2　対象ユーザ

2　　本ソフトウェアは MeeGo を搭載したタブレットを持つユーザを対象としている。

2.1　記載範囲

本設計書ではデータ共有システムが搭載する機能について述べる。

2.2　参照ドキュメント

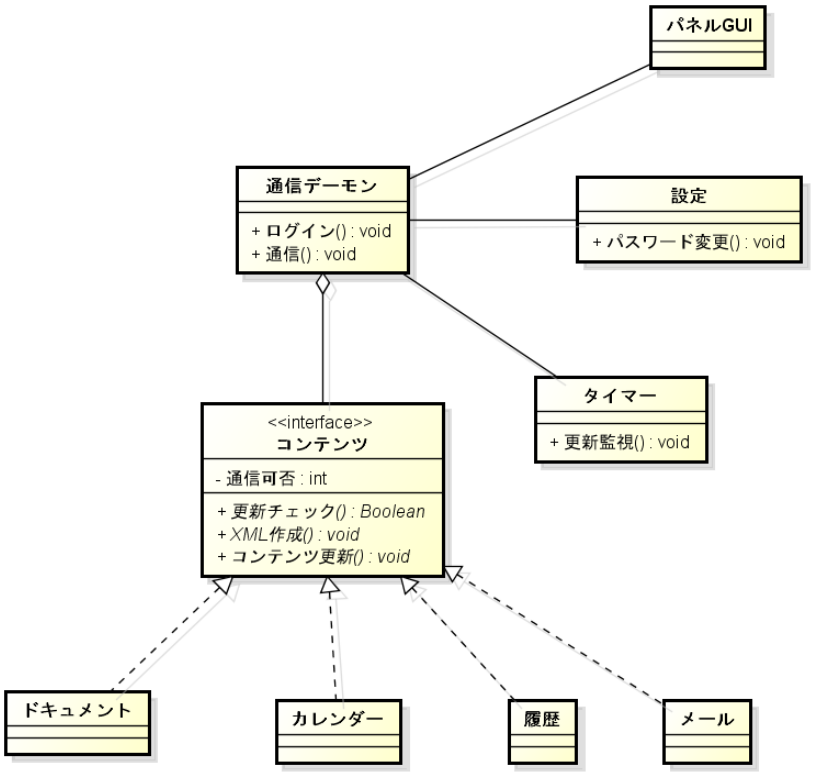
ID	文書名	文書番号	発行年月	備考
	要件定義書			

2.3　定義（用語、略語）

ID	用語・略号	正式表記	意味

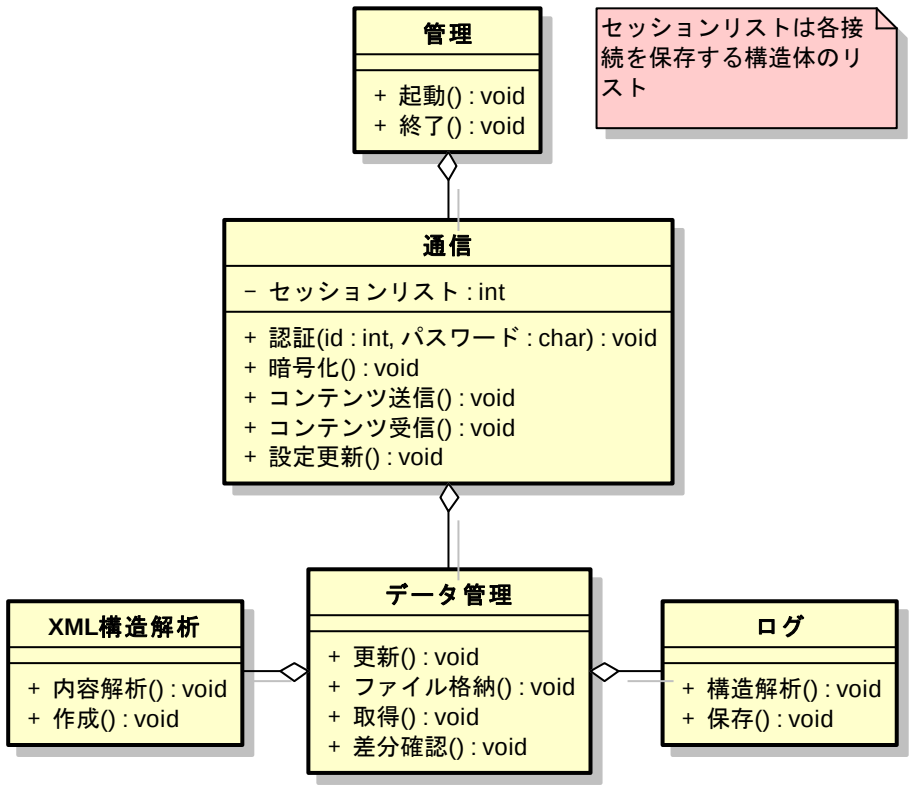
3　システム構成

クライアントソフトウェア



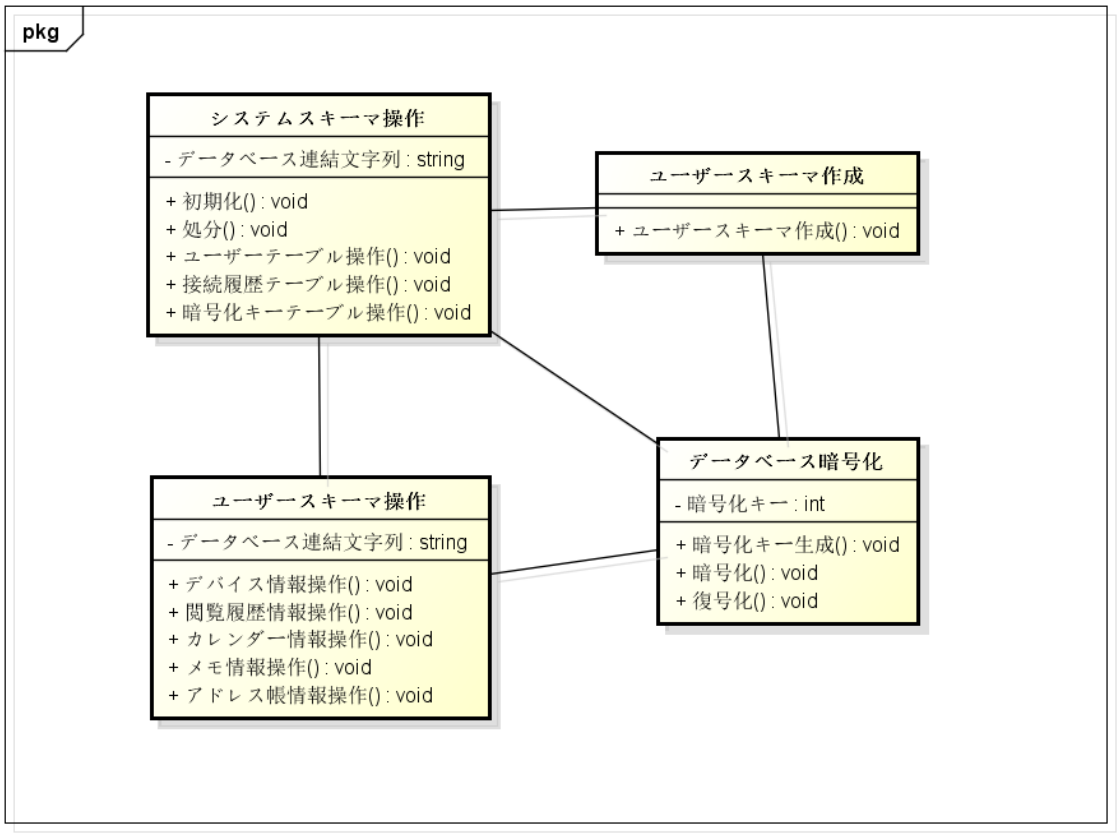
クライアントソフトウェアの分析クラス図を示す。クライアントソフトウェアは画面から入力を受け取り、サーバへの通信を開始する。また、ユーザからの入力によりパスワードの変更などの設定を行うことも可能とする。通信を開始した後はタイマーモジュールで 30 秒周期で各データが更新されているかどうかをチェックし、それぞれの共有項目ごとに XML の作成を行う。また通信デーモンは接続後はサーバからの通信を待ち、データを受信した場合は各項目に更新を行わせる。

サーバーソフトウェア



サーバーソフトウェアの分析クラス図を示す。サーバーソフトウェア起動後、クライアントからの接続を監視する。クライアントからの接続要求が来たら、セキュリティを行なう。
クライアントから受信したデータに対して、命令解析を行ってから、データの処理（DB 更新、他のクライアントへの送信）を行う。

データベース操作モジュール



データベース操作モジュールの分析クラス図を示す。データベース操作モジュールでは、サーバーソフトウェアからの要求を受け取り、要求によって、データベースの操作を行う、また、非機能要件を満たすために、データベースの暗号化を行う。

4　機能概要

4.1　システムを構成する機能

クライアントソフトウェア

機能 ID	機能名	機能概要	役割分担	関連機能 ID
001	ログイン機能	サーバへの通信を開始する	許	
002	通信機能	データの送受信の制御を行う	許	
003	設定機能	パスワードの再設定を行う	許	
004	比較機能	一定時間ごとに各共有機能の更新監視機能を起動する	許	
005	XML 生成・解析機能	アプリケーションデータから XML の生成、XML からアプリケーションデータの生成を行う	許、森本、劉建宇	
006	Contact 共有機能	Contact 情報について更新監視、データ更新を行う	許	
007	Note 共有機能	Note 情報について更新監視、データ更新を行う	森本	
008	UI 機能	ログインや新規登録を行うためのユーザー操作画面を提供する	森本	

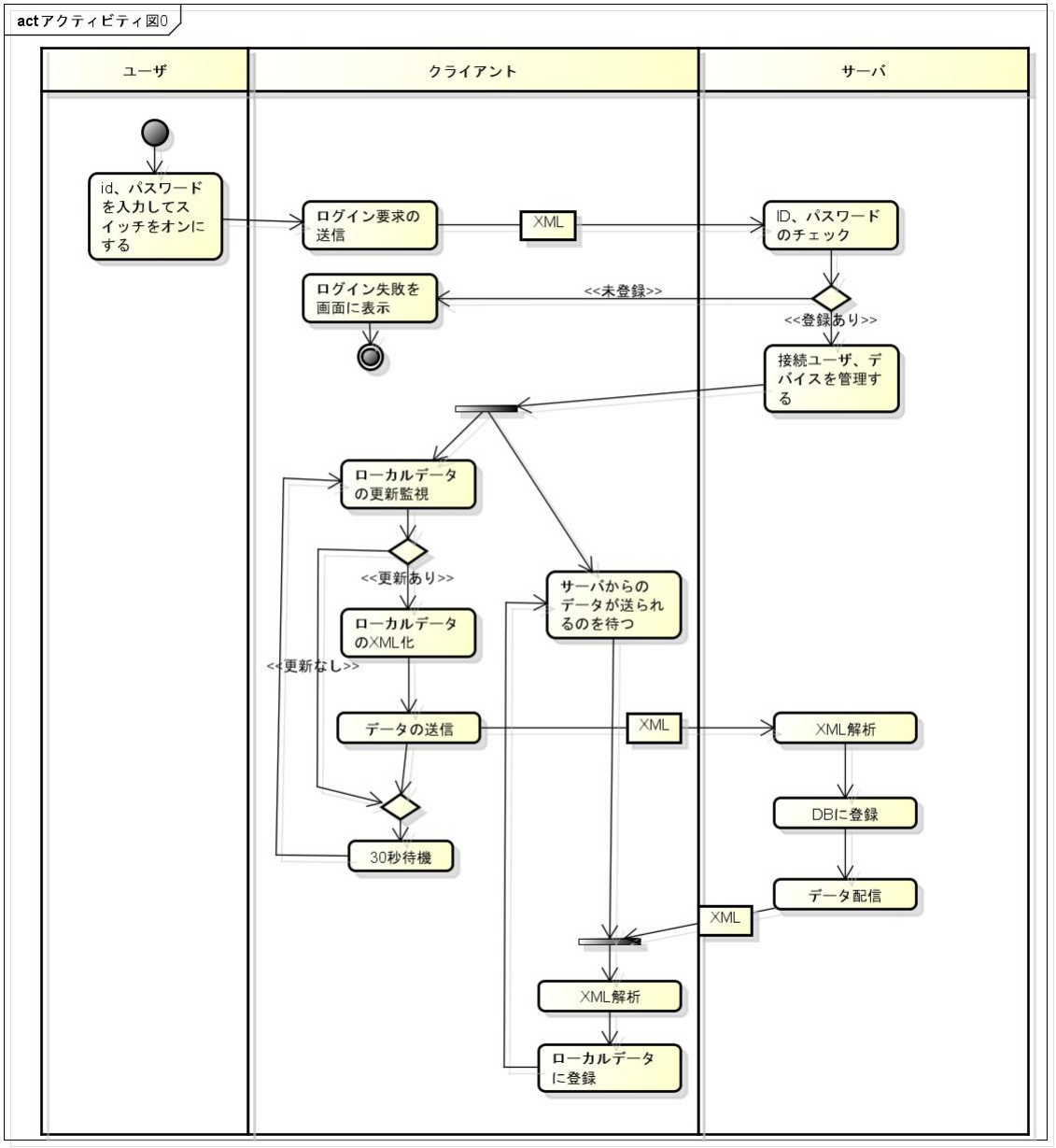
サーバーソフトウェア

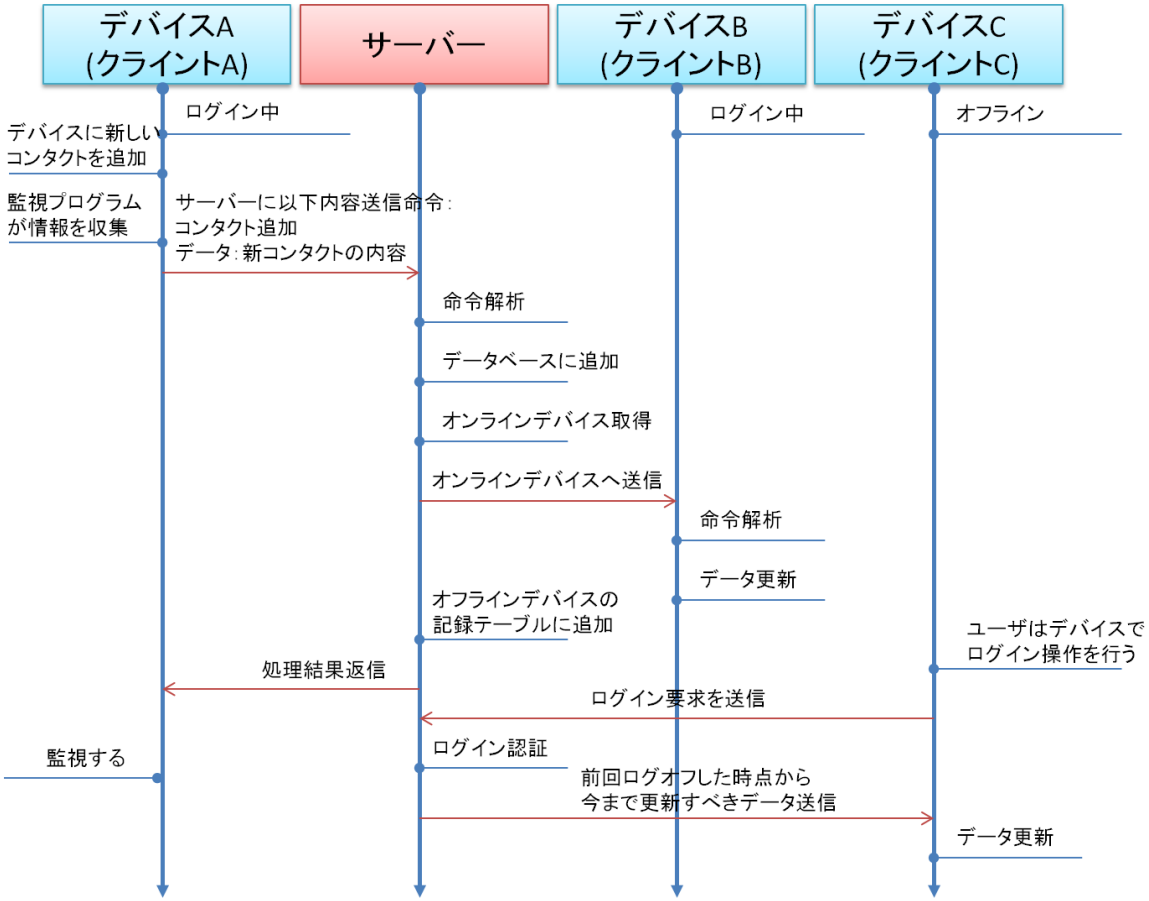
機能 ID	機能名	機能概要	役割分担	関連機能 ID
	通信機能	データの送受信の制御を行う	劉宏	
	接続管理	クライアントの接続を管理する	劉宏	
	XML 生成・解析機能	アプリケーションデータから XML の生成、XML からアプリケーションデータの生成を行う	劉宏	

	セキュリティ	通信時のセキュリティを行う	劉宏	
	システムスキーマ操作機能	システムスキーマのテーブルを操作する	劉建	
	ユーザスキーマ操作機能	ユーザースキーマのテーブルを操作する	劉建	
	ユーザースキーマ作成機能	ユーザーのスキーマとテーブルを作成する	劉建	
	データベース暗号化機能	データベースのデータを暗号・復号化する	劉建	

4.2　処理の流れ

サーバ・クライアント間の処理の流れを以下に示す


actアクティビティ図0
ユーザ
クライアント
サーバ
id、パスワードを入力してスイッチをオンにする
ログイン要求の送信
XML
ID、パスワードのチェック
ログイン失敗を画面に表示
<<未登録>>
<<登録あり>>
接続ユーザ、デバイスを管理する
ローカルデータの更新監視
<<更新あり>>
ローカルデータのXML化
<<更新なし>>
サーバからのデータが送られるのを待つ
データの送信
XML
XML解析
DBに登録
データ配信
30秒待機
XML
XML解析
ローカルデータに登録


デバイスA
(クライントA)
サーバー
デバイスB
(クライントB)
デバイスC
(クライントC)
ログイン中
ログイン中
オフライン
デバイスに新しい
コンタクトを追加
監視プログラム
が情報を収集
サーバーに以下内容送信命令:
コンタクト追加
データ:新コンタクトの内容
命令解析
データベースに追加
オンラインデバイス取得
オンラインデバイスへ送信
命令解析
データ更新
オフラインデバイスの
記録テーブルに追加
ユーザはデバイスで
ログイン操作を行う
処理結果返信
ログイン要求を送信
ログイン認証
監視する
前回ログオフした時点から
今まで更新すべきデータ送信
データ更新

5　機能詳細

機能 ID	001
機能名	ログイン機能
概要	
サーバへのログインを行う	
機能詳細	
サーバにソケット通信で接続のリクエストを送る。 接続が確立された後に画面に入力されたユーザ ID、パスワードを用いてサーバへログインを要求する XML を作成し、サーバに送信する。	
関連機能	

機能 ID	002
機能名	通信機能
概要	
データの送受信の制御を行う	
機能詳細	
接続されたソケットを利用し、サーバからデータが送られてくるのを待つ。サーバからデータを受信すると、XML 解析を行って各共有機能が持つデータ更新処理を呼び出す。 また、更新監視機能によりデータの更新があったことが分かった場合、XML 化されたデータを受け取ってサーバにそれらを送信する。	
関連機能	

機能 ID	003
機能名	設定機能
概要	
パスワードの設定を行う	
機能詳細	
画面に入力されたアカウント名、既存のパスワード、新規パスワードを使って XML を作成し、サーバにパスワードの変更処理を要求する。	
関連機能	

機能 ID	004
機能名	比較機能
概要	
一定時間ごとに各共有機能の更新監視機能を起動する	
機能詳細	

タイマーを起動し、30 秒周期で各共有機能が持つ比較処理を起動する。 ここで新しいデータがあった場合、XML 生成機能により XML を生成し、通信機能を通してサーバにデータを送信する。	
関連機能	

機能 ID	005
機能名	XML 生成・解析機能
概要	
アプリケーションデータから XML の生成、XML からアプリケーションデータの生成を行う	
機能詳細	
共有項目ごとにデータを受け取りサーバが解釈できる形の XML に変換する。 また、XML を受け取り、各共有項目を更新するためのデータを取り出す。	
関連機能	

機能 ID	006
機能名	Contact 共有機能
概要	
MeeGo コンタクトデータの取得や更新を実現する。	
機能詳細	
コンタクトでデータの追加、変更、削除を行った場合はそのデータを取得する。 逆に追加や変更、削除すべきデータがあれば、そのデータをコンタクトには反映させる。	
関連機能	

機能 ID	007
機能名	Note 共有機能
概要	
Note 情報について更新監視、データ更新を行う	
機能詳細	
データベースに更新ログを取得するためのテーブルを作成し、比較機能から呼び出される度にテーブルを確認して更新されたデータを取得する。 また、サーバから受け取ったデータに基づき、新たなノートを作成してデータベースに登録する	
関連機能	

機能 ID	008
機能名	UI 機能
概要	

ログインや新規登録を行うためのユーザー操作画面を提供する
機能詳細
ログインや新規登録を行うためのユーザー操作画面を提供する
関連機能

機能 ID	
機能名	通信機能（サーバー）
概要	
データの送受信の制御を行う	
機能詳細	
セッションごともつ機能である。 接続されたソケットを利用し、クライアントからデータが送られてくるのを待つ。クライアントからデータを受信すると、XML 解析を行ってデータ処理を行う。 また、他のデバイスへ送信が必要になる場合、他のセッションの通信メソッドを呼ぶ出す。	
関連機能	

機能 ID	
機能名	接続管理
概要	
接続中のクライアントの接状況の管理を行う。	
機能詳細	
サーバープログラムは集中的にクライアントの接続状況を管理する。 何かの原因でクライアントの接続が切れた場合、ユーザがログオフした場合、ユーザログインした場合など、接続の作成、削除などの処理を行う。	
関連機能	

機能 ID	
機能名	XML 生成・解析機能
概要	
クライアントから XML の解析、クライアントへ送信用 XML データの生成を行う。	
機能詳細	
基本的には、.netFramework の XML 操作クラスを利用する。 XML 解析した命令でデータ処理を行う。	
関連機能	

機能 ID	
機能名	セキュリティ機能

概要	
通信時のセキュリティを行う処理である。	
機能詳細	
クライアントからの接続要求が来た場合行う。 以下2つ処理が行われる。 ・サーバー認証 ・通信データの暗号化と復号化	
関連機能	

機能 ID	
機能名	システムスキーマを操作機能
概要	
システムスキーマのテーブルを操作する	
機能詳細	
サーバープログラムの要求に応じて、データベースのテーブル操作を行う。 以下3つのテーブル処理が行われる。 ・ユーザー情報 ・システムに接続情報 ・ユーザーに属する暗号化キー情報	
関連機能	

機能 ID	
機能名	ユーザースキーマを操作機能
概要	
ユーザースキーマのテーブルを操作する	
機能詳細	
サーバープログラムの要求に応じて、データベースのテーブル操作を行う。 以下5つの情報処理が行われる。 ・デバイス情報 ・コンタクト情報 ・メモ情報 ・閲覧履歴情報 ・カレンダー情報	
関連機能	

機能 ID	
機能名	ユーザースキーマを作成機能

概要
ユーザーのスキーマとテーブルを作成する
機能詳細
ユーザーが新規登録する時、ユーザーのスキーマとテーブルを作成する。 また、ユーザー情報をユーザーテーブルに追加する。
関連機能	

機能 ID	
機能名	データベース暗号化機能
概要	
データベースのデータを暗号・復号化する。	
機能詳細	
サーバーからの要求が来た場合行う。 以下2つ処理が行われる。 ・暗号化ためのキーを作成する ・データの暗号化と復号化	
関連機能	

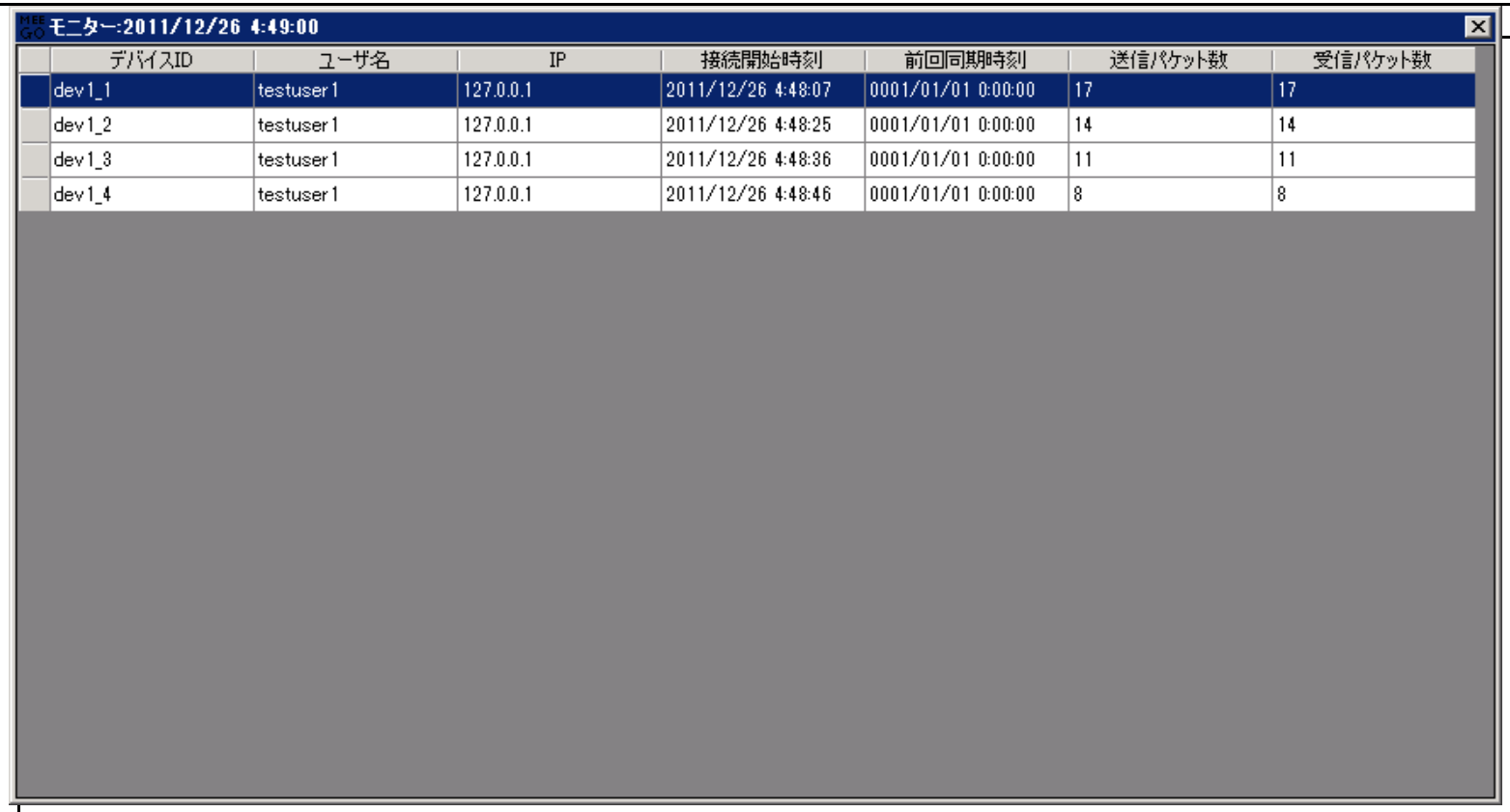
ID	namespace	クラス名	分類	名前	機能	戻り値	パラメータ
1		SystemDBOperator	接続関係	int Initiate(string connString)	データベース接続 open	int 0：ok 0以外：エラーコード	connString　型：string
2				void Dispose()	終了処理 DBConnection closeなど	なし	なし
3			ユーザ関係	int AddUser(string userName,string password,string mail,string deviceId)	新しいユーザを登録する ※DB作成も含む	int 0：ok 0以外：エラーコード	userName　型：string password　型：string deviceId　型：string
4				int GetUsers(out List<string> userNames)	既存ユーザを取得する	int 0：ok 1以外：エラーコード	userNames　型：List
5				int DeleteUser(string userName)	ユーザ削除（退会時）使用 ※DB削除も含む	int 0：ok 1以外：エラーコード	userName　型：string
6				int ChangePassword(string userName,string newpassword)	パスワード変更する	int 0：ok 0以外：エラーコード	userName　型：string newpassword　型：string
7				UserDBOperator GetUserDBOperator(UserName)	ユーザ名対応DBしかアクセスできないUserDBOperatorインスタンスを取得する		userName　型：string
8				int UserLogon(string userName,string password,out bool result,string deviceId,out bool isNewDevice)	ユーザログイン	int 0：ok 0以外：エラーコード	userName　型：string newpassword　型：string result　型：bool（0:失敗　1:成功） deviceId　型：string isNewDevice　型：bool（0:No 1:Yes）
			接続履歴関係	int AddConnectionLog(string userName,string deviceid,DateTime logOnTime)	ユーザの接続履歴を追加する	int 0：ok 0以外：エラーコード	userName　型：string deviceId　型：string logOnTime　型：DateTime
				int GetConnectionLog(string userName,string deviceId,out Datetime logOnTime,out DateTime logOffTime)	ユーザの接続履歴を取得する	int 0：ok 0以外：エラーコード	userName　型：string deviceId　型：string logOnTime　型：DateTime logOffTime　型：DateTime
				int UpdateLogOffTime(string userName,string deviceid,DateTime logoffTime)	ユーザのログオフ時間を更新する	int 0：ok 0以外：エラーコード	userName　型：string deviceId　型：string logOffTime　型：DateTime
				int DeleteConnectionLog(string userName,string deviceId)	ユーザーの接続履歴を削除	int 0：ok 0以外：エラーコード	userName　型：string deviceId　型：string
9			接続関係	UserDBOperator(string connString, string UserName)		なし	connString　型：string userName　型：string
10				int Open()	初期処理 userNameでDB選択 DBConnection開始など ※userName指定後、このユーザのDBだけアクセスするようにODBCだと、ChangeDatabase	int 0：ok 0以外：エラーコード	
11				void Dispose()	終了処理 DBConnection closeなど	なし	なし
12			デバイス関係	int GetDeviceIds(out List<string> deviceIds)	deviceIDを全部取得する	int 0：ok 0以外：エラーコード	deviceIds　型：list
13				int GetOnlineDeviceIds(out List<string> onLineDeviceIds)	オンラインdeviceIDを取得する	int 0：ok 0以外：エラーコード	onLineDeviceIds　型：list
14				int GetOfflineDeviceIds(out List<string> offLineDeviceIds)	オフラインdeviceIDを取得する	int 0：ok 0以外：エラーコード	offLineDeviceIds　型：list
15				int GetSyncProperty(string deviceId, out bool contacts,out bool calendar,out bool accessHistory)	deviceごとの同期項目を取得する	int 0：ok 0以外：エラーコード	deviceId　型：string contacts　型：bool calendar　型：bool accessHistory　型：bool
16				int UpdateDeviceState (string deviceId, bool deviceState)	deviceの接続状態を更新する	int 0：ok 0以外：エラーコード	deviceId　型：string deviceState　型：bool

17				int SetSyncProperty(string deviceId, bool contacts,bool calendar,bool accessHistory)	deviceごとの同期項目を設定する	int 0:ok 0以外:エラーコード	deviceId 型:string contacts 型:bool calendar 型:bool accessHistory 型:bool
18			コンタクト関係	int GetContactLastUpdateTime(string contactId,out Datetime lastUpdateTime)	あるコンタクト前回更新した時間を取得する どちが最新を比較する時使用	int 0:ok 0以外:エラーコード	contactId 型:string lastUpdateTime 型:DateTime
19				int GetAllContacts(out DataTable contacts)	新ディバイス対応 ユーザコンタクトを全部取得	int 0:ok 0以外:エラーコード	(column:contactId,contactContents,lastUpdateTime)
20				int AddNewContact(string contactId,string contactContents,Datetime lastUpdateTime)	コンタクトを追加する	int 0:ok 0以外:エラーコード	contactId 型:string contactContents 型:string lastUpdateTime 型:DateTime
21				int UpdateContact(string contactId,string contactContents,Datetime lastUpdateTime)	コンタクトを変更する	int 0:ok 0以外:エラーコード	contactId 型:string contactContents 型:string lastUpdateTime 型:DateTime
22				int DeleteContact(string contactId)	コンタクトを無効する	int 0:ok 0以外:エラーコード	contactId 型:string contactContents 型:string
23			コンタクトスタック関係	int AddContactToStack(string deviceId,string contactId)	オフラインデバイスの更新すべきcontactidを追加する	int 0:ok 0以外:エラーコード	deviceId 型:string contactId 型:string
24				int GetContactStackData(string devicdId, out DataTable contacts)	デバイスがstackデータを取得 ※stackテーブルとcontactテーブルの結合処理は必要	int 0:ok 0以外:エラーコード	deviceId 型:string contacts 型:DataTable (column:contactId,contactContents,lastUpdateTime)
25				int DeleteContactStack(string deviceId)	デバイスに「11番」で取ったデータを発送成功した時点呼び出す	int 0:ok 0以外:エラーコード	deviceId 型:string
26	DBAccessor		カレンダー関係	int GetCalendarLastUpdateTime(string calendarId,out Datetime lastUpdateTime)	あるカレンダーイベント前回更新した時間を取得する どちが最新を比較する時使用	int 0:ok 0以外:エラーコード	calendarId 型:string lastUpdateTime 型:DateTime
27				int GetAllCalendars(out DataTable calendars)	新ディバイス対応 カレンダーイベントを全部取得	int 0:ok 0以外:エラーコード	(column:calendarId ,calendarContents,lastUpdateTime)
28				int AddNewCalendar(string calendartId,string calendarContents,Datetime lastUpdateTime)	カレンダーイベントを追加する	int 0:ok 0以外:エラーコード	calendarId 型:string calendarContents 型:string lastUpdateTime 型:DateTime
29				int UpdateCalendar(string calendarId,string calendarContents,Datetime lastUpdateTime)	カレンダーイベントを変更する	int 0:ok 0以外:エラーコード	calendarId 型:string calendarContents 型:string lastUpdateTime 型:DateTime
30				int DeleteCalendar(string calendarId)	カレンダーイベントを無効する	int 0:ok 0以外:エラーコード	calendarId 型:string
31			カレンダースタック関係	int AddCalendarToStack(string deviceId,string calendarId)	オフラインデバイスの更新すべきカレンダーイベントを追加する	int 0:ok 0以外:エラーコード	deviceId 型:string calendarId 型:string
32				int GetCalendarStackData(string devicdId, out DataTable calendars)	デバイスがstackデータを取得 ※stackテーブルとカレンダーテーブルの結合処理は必要	int 0:ok 0以外:エラーコード	deviceId 型:string calendars 型:DataTable (column:calendarId ,calendarContents,lastUpdateTime)
33				int DeleteCalendarStack(string deviceId)	デバイスに「11番」で取ったデータを発送成功した時点呼び出す	int 0:ok 0以外:エラーコード	deviceId 型:string
34			閲覧履歴関係	int GetHistoryUpdateTime(string historyId,out Datetime lastUpdateTime)	ある閲覧履歴前回更新した時間を取得する どちが最新を比較する時使用	int 0:ok 0以外:エラーコード	historyId 型:string lastUpdateTime 型:DateTime
35				int GetAllHistorys(out DataTable historys)	新ディバイス対応 閲覧履歴を全部取得	int 0:ok 0以外:エラーコード	(column:historyId ,historyContents,lastUpdateTime)
36				int AddNewHistory(string historyId,string historyContents,Datetime lastUpdateTime)	閲覧履歴を追加する	int 0:ok 0以外:エラーコード	historyId 型:string historyContents 型:string lastUpdateTime 型:DateTime
37				int UpdateHistory(string historyId,string historyContents,Datetime lastUpdateTime)	閲覧履歴を変更する	int 0:ok 0以外:エラーコード	historyId 型:string historyContents 型:string lastUpdateTime 型:DateTime

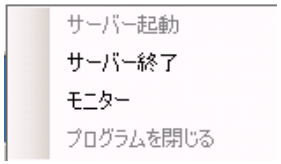
UserDBOperator

38			int DeleteHistory(string historyId)	閲覧履歴を無効する	int 0:ok 0以外:エラーコード	historyId 型:string
39		閲覧履歴 スタック関係	int AddHistoryToStack(string deviceId,string historyId)	オフラインデバイスの更新すべく閲覧履歴を追加する	int 0:ok 0以外:エラーコード	deviceId 型:string historyId 型:string
40			int GetHistoryStackData(string devicdId, out DataTable historys)	デバイスがstackデータを取得 ※stackテーブルと閲覧履歴テーブルの結合処理は必要	int 0:ok 0以外:エラーコード	deviceId 型:string historys 型:DataTable (column:historyId, historyContents,lastUpdateTime)
41			int DeleteHistoryStack(string deviceId)	デバイスに「１１番」で取ったデータを発送成功した時点呼び出す	int 0:ok 0以外:エラーコード	deviceId 型:string
42		ノートブック関係	int GetNoteBookLastUpdateTime(string noteBookId,out Datetime lastUpdateTime)	あるノートブック前回更新した時間を取得する どちが最新を比較する時使用	int 0:ok 0以外:エラーコード	noteBookId 型:string lastUpdateTime 型:DateTime
43			int GetAllNoteBooks(out DataTable noteBooks)	新ディバイス対応 ノートブックを全部取得	int 0:ok 0以外:エラーコード	(column:noteBookId ,noteBookContents,lastUpdateTime)
			int AddNewNoteBook(string noteBookId,string noteBookContents,Datetime lastUpdateTime)	ノートブックを追加する	int 0:ok 0以外:エラーコード	noteBookId 型:string noteBookContents 型:string lastUpdateTime 型:DateTime
			int UpdateNoteBook(string noteBookId,string noteBookContents,Datetime lastUpdateTime)	ノートブックを変更する	int 0:ok 0以外:エラーコード	noteBookId 型:string noteBookContents 型:string lastUpdateTime 型:DateTime
			int DeleteNoteBook(string noteBookId)	ノートブックを無効する	int 0:ok 0以外:エラーコード	noteBookId 型:string
		ノートブック スタック関係	int AddNoteBookToStack(string deviceId,string noteBookId)	オフラインデバイスの更新すべきノートブックを追加する	int 0:ok 0以外:エラーコード	deviceId 型:string noteBookId 型:string
			int GetNoteBookStackData(string devicdId, out DataTable noteBooks)	デバイスがstackデータを取得 ※stackテーブルとノートブックテーブルの結合処理は必要	int 0:ok 0以外:エラーコード	deviceId 型:string noteBooks 型:DataTable (column:noteBookId ,noteBookContents,lastUpdateTime)
			int DeleteNoteBookStack(string noteBookId)	デバイスに「１１番」で取ったデータを発送成功した時点呼び出す	int 0:ok 0以外:エラーコード	deviceId 型:string
		ノート関係	int GetNoteLastUpdateTime(string noteId,out Datetime lastUpdateTime)	あるノート前回更新した時間を取得する どちが最新を比較する時使用	int 0:ok 0以外:エラーコード	noteId 型:string lastUpdateTime 型:DateTime
			int GetAllNotes(out DataTable notes)	新ディバイス対応 ノートを全部取得	int 0:ok 0以外:エラーコード	(column:noteId ,noteContents,lastUpdateTime)
			int AddNewNote(string noteId,string noteContents,Datetime lastUpdateTime)	ノートを追加する	int 0:ok 0以外:エラーコード	noteId 型:string noteContents 型:string lastUpdateTime 型:DateTime
			int UpdateNote(string noteId,string noteContents,Datetime lastUpdateTime)	ノートを変更する	int 0:ok 0以外:エラーコード	noteId 型:string noteContents 型:string lastUpdateTime 型:DateTime
			int DeleteNote(string noteId)	ノートを無効する	int 0:ok 0以外:エラーコード	noteId 型:string
		ノート スタック関係	int AddNoteToStack(string deviceId,string noteId)	オフラインデバイスの更新すべきノートを追加する	int 0:ok 0以外:エラーコード	noteId 型:string noteId 型:string
			int GetNoteStackData(string devicdId, out DataTable notes)	デバイスがstackデータを取得 ※stackテーブルとノートテーブルの結合処理は必要	int 0:ok 0以外:エラーコード	noteId 型:string notes 型:DataTable (column:calendarId ,calendarContents,lastUpdateTime)
			int DeleteNoteStack(string deviceId)	デバイスに「１１番」で取ったデータを発送成功した時点呼び出す	int 0:ok 0以外:エラーコード	noteId 型:string

画面レイアウト	画面名	モニタリング画面	作成日	2011/12/15	作成者	劉宏超
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					



画面レイアウト	画面名	タスクバー画面	作成日	2011/12/15	作成者	劉宏超
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					



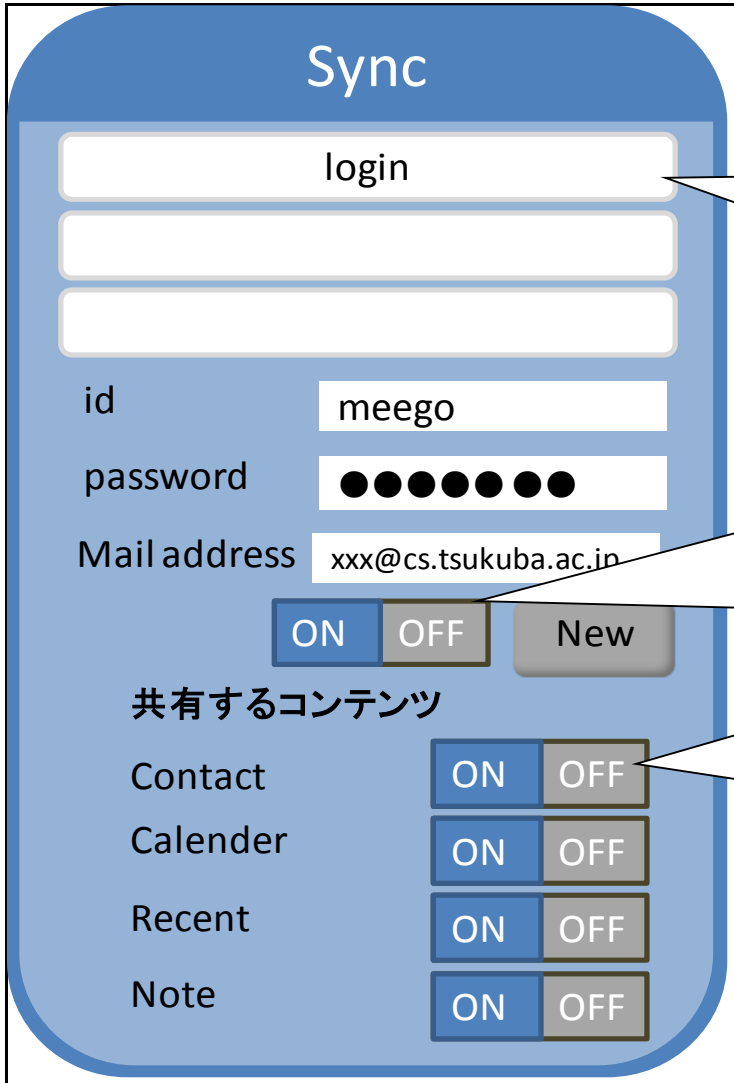
画面定義書	画面名	モニタリング画面	作成日	2011/12/15	作成者	劉宏超
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					

項番	項目名	属性						チェックパターン	入力	備考
		全角/半角	数字	英数	カナ	漢字	特殊			
1	タイトル	○	○		○		○	−	−	モニター:[システム時間]
2	デバイスID			○				−	−	
3	ユーザ名			○				−	−	
4	IPアドレス		○				○	−	−	NNN.NNN.NNN.NNN
5	接続開始時刻		○				○	−	−	YYYY/MM/DD H:MM:SS
6	前回同期時刻		○				○	−	−	YYYY/MM/DD H:MM:SS
7	送信パケット数		○					−	−	
8	受信パケット数		○					−	−	

画面定義書	画面名	タスクバー画面	作成日	2011/12/15	作成者	劉宏超
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					

項番	項目名	属性						チェックパターン	入力	備考
		全角/半角	数字	英数	カナ	漢字	特殊			
1	サーバー起動ボタン	−	−	−	−	−	−	−	○	
2	サーバー終了ボタン	−	−	−	−	−	−	−	○	
3	モニターリングボタン	−	−	−	−	−	−	−	○	
4	プログラム終了ボタン	−	−	−	−	−	−	−	○	

画面レイアウト	画面名	パネル表面	作成日	2011/12/1	作成者	森本
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					



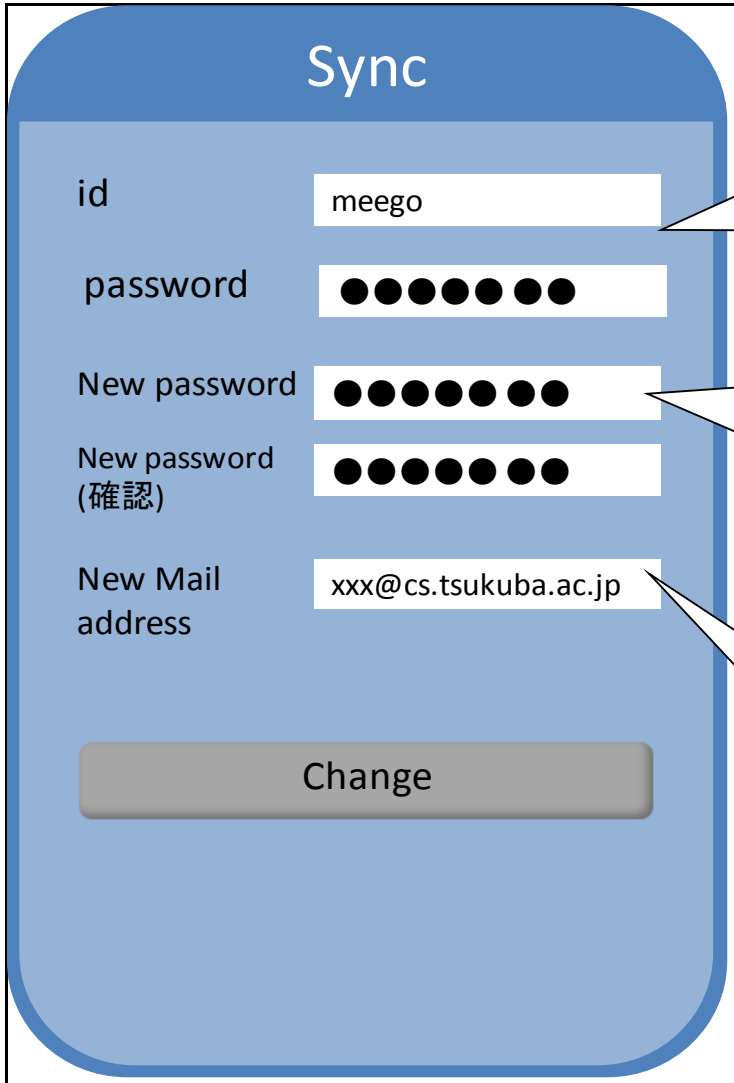
login、logout、login errorなどのステータスを直近三回分表示する

スイッチがオンに切り替わった時にid、passwordを引数にpluginMainクラスのturnSwitchOnメソッドを呼び出し、サーバと通信を開始する。
スイッチがオフに切り替わった際にturnSwitchOffメソッドを呼び出し通信を終了する

スイッチがオンに切り替わった際に各共有モジュールのAllowUpdate変数を1にする。また、offに切り替わった際にAllowUpdate変数を0にする

Reference
●MeeGoCloud_morimoto.asta

画面レイアウト	画面名	パネル裏面	作成日	2011/12/1	作成者	森本
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					



現在のpasswordが未入力の場合、password、Mail addressの変更は適用せずにpasswordを入力する旨を表示する

新しいpasswordと確認のpasswordが異なる場合は変更を適用せず、確認用のpasswordが異なる旨をダイアログで表示する

空欄の場合はメールアドレスの変更は行わない。また、メールアドレスが有効かどうかのチェックは行わない

画面定義書	画面名	パネル表面	作成日	2012/1/7	作成者	森本
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					

項番	項目名	属性						チェックパターン	入力	備考
		全角/半角	数字	英数	カナ	漢字	特殊			
1	メッセージボックス1		○	○	○	○	○	－	－	
2	メッセージボックス2		○	○	○	○	○	－	－	
3	メッセージボックス3		○	○	○	○	○	－	－	
4	ID入力ボックス	半角	○	○				10文字以下	○	
5	assword入力ボック	半角	○	○				10文字以下	○	入力文字列を隠す
6	il Address入力ボッ	半角	○	○			○	－	○	
7	通信開始スイッチ								○	
8	アカウント登録ボタン								○	
9	Contact共有スイッチ								○	
10	Calendar共有スイッチ								○	
11	Recent共有スイッチ								○	
12	Note共有スイッチ								○	

画面定義書	画面名	パネル裏面	作成日	2012/1/7	作成者	森本
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					

項番	項目名	属性						チェックパターン	入力	備考
		全角/半角	数字	英数	カナ	漢字	特殊			
1	ID入力ボックス	半角	○	○				10文字以下	○	
2	asswordl入力ボックス	半角	○	○				10文字以下	○	入力文字列を隠す
3	password入力ボック	半角	○	○				10文字以下	○	入力文字列を隠す
4	用password入力ボッ	半角	○	○				10文字以下	○	入力文字列を隠す
5	ail Address入力ボッ	半角	○	○			○	－	○	
6	変更実行ボタン								○	

文書名

【ソフトウェア詳細設計書（クライアント側）】

文書番号

04-001

	作成
作成者名	許　先華　森本悠矢
作成日	2011 年 10 月 2 日

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考
1	2012/01/14	V1.1	設計クラス図	
2	2012/01/15	V1.2	設計クラス図・Note，Contact，Calendar のクラス詳細追加	

目　次

1　概要

1.1　目的

本書はデータ共有システムクライアント側のソフトウェア設計仕様について記述する

1.2　方針

初めて MeeGo デバイスをベースにしたソフトウェア開発であるため、下記の点に重点を置きった。

1. 要求定義書と基本設計書に定めたことを基づいて、設計を行う。
2. MeeGo はオーペンソースであるため、そのドキュメントやサンプルプログラムをより理解し、参考にする。

1.3　記載範囲

本書はクライアント側のソフトウェア構成、インタフェース、機能仕様を記載する

1.4　参照ドキュメント

ID	文書名	文書番号	発行年月	備考
01_001	要件定義書			
02_001	基本設計書			

1.5　定義（用語、略語）

ID	用語･略号	正式表記	意味
01	コンタクトデータ		Contact アプリケーションで使用されているデータ
02	ノートデータ		Note アプリケーションで使用されているデータ
03	履歴データ		Web の訪問履歴のデータ
04			

2　クラス構造

2.1　設計クラス図

分析ラクスを基に、設計クラス図を作成した。各クラスの詳細は別の表に説明する。

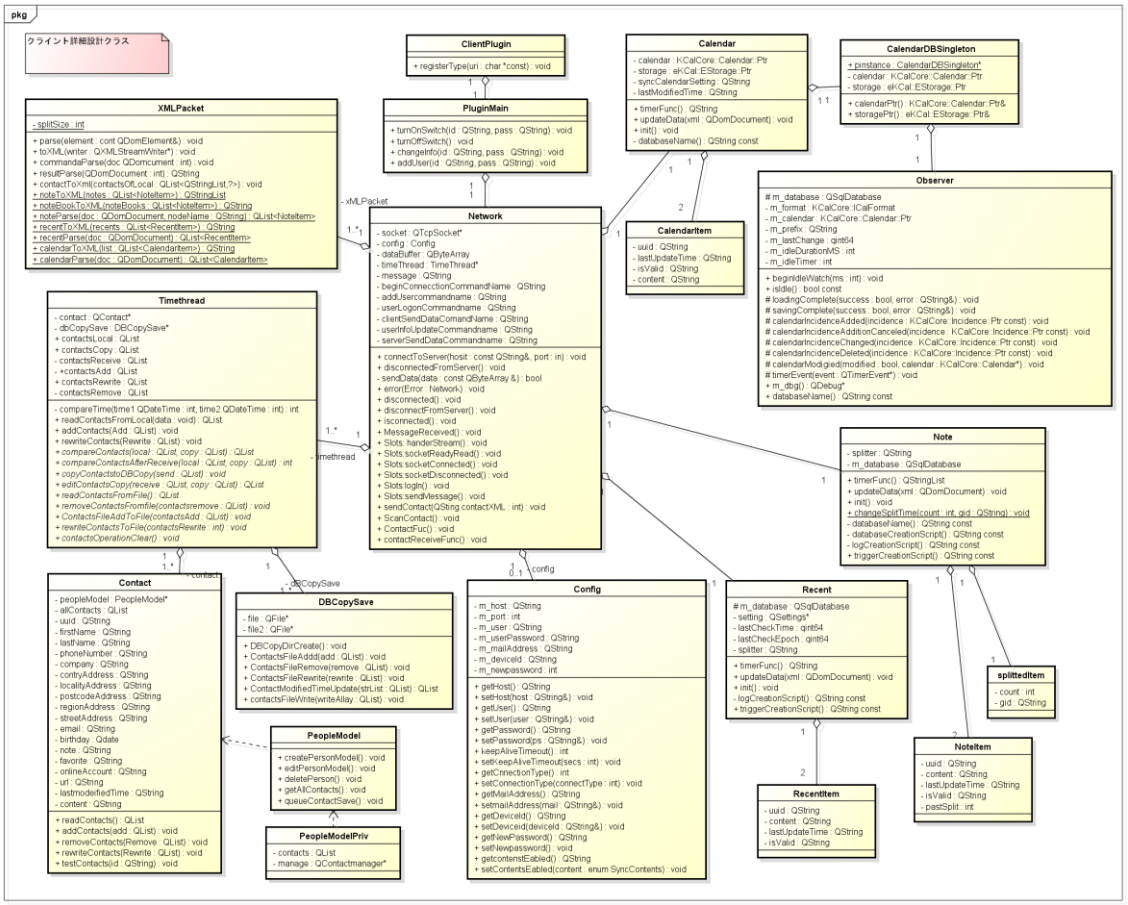


図 1 設計クラス図

2.2 PluginMain クラス詳細

クラス詳細				
クラス名	PluginMain			
親クラス	なし			
可視性	Public			
クラス概要				
UI から呼び出す処理を定義する				
インクルード				
1	QString			
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	turnOnSwitch		QString,QString	ログインの処理を行う
public	turnOffSwitch			ログオフの処理を行う
public	changeInfo		QString,QStri	パスワードの変更処理を行

			ng	う
public	addUser		QString,QString	ユーザの新規登録処理を行う
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他

2.3 Contact クラスの詳細

クラス詳細				
クラス名	Contact			
親クラス	なし			
可視性	Public			
クラス概要				
コンタクト同期に完成操作				
インクルード				
1	QContactAddress		7	QContactOrganization
2	QContactAvatar		8	QContactTimestamp
3	QContactBirthday		9	QContactNote
4	QContactEmailAddress		10	QContactPhoneNumber
5	QContactGuid		11	peoplemodel.h
6	QContactName		12	Peoplemodel_p.h
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	readContacts	QList		コンタクトからデータを読み取る
public	addContacts	void	QList	コンタクトにデータを追加
Public	removeContacts	void	QList	指定されたデータを削除
Public	rewriteContacts	void	QList	指定されたデータを編集
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	peopleModel	PeopleModel*		
Private	allCtoancts	QList		
Private	Uuid	QString		
Private	firstName	QString		
Private	lastName	QString		
Private	phoneNumber	QString		

Private	Company	QString		
Private	Email	QString		
Private	birthday	QDate		
Private	note	QString		
Private	lastmodifiedTime	QDateTime		
Private	content	QString		

2.4 DBCopySave ラスの詳細

クラス詳細				
クラス名	DBCopySave			
親クラス	なし			
可視性	Public			
クラス概要				
ファイル ContactCopyFile.txt のデータを取得、更新				
インクルード				
1	QFile		7	
2	QDir		8	
3	QIODevice		9	
4	QTime		10	
5	QDebug		11	
6	QVector		12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	DBCopyDirCreate	void		ContactCopyFile の Dir を作成
public	ContactsFileAdd	void	QList	ファイルにデータを追加
Public	ContactsFileRewrite	void	QList	ファイルにデータを編集
Public	ContactsFileRemove	void	QList	ファイルにデータを削除
Public	ContactsModifiedTimeUpdate	QLsit	QList	データの更新時間を変更
Public	ContactsFileWrite	Void	QList	ファイルにデータを書き込む
Public	ContactsFileRead	QList		ファイルにデータを書き込む
	DBCopySave			コンストラクタ
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	file	QFile*		ファイルを読むためのインスタンス
private	file2	QFile*		ファイルを書くためのインスタ

				ンス

2.5 Note クラス詳細

クラス詳細				
クラス名	Note			
親クラス	なし			
可視性	Public			
クラス概要				
Note データが格納されているデータベース（note.db）に接続し、更新されたかどうかの監視とデータを受信した場合の更新処理を行う				
インクルード				
1	QObject		7	QVariant
2	QtSql/QSqlDatabase		8	QUuid
3	QtSql/QSqlQuery		9	QDomDocument
4	QtSql/QSqlRecord		10	xmlpacket.h
5	QSqlError		11	noteItem.h
6	QDir		12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	timerFunc	QString		データベースにアクセスし、変更されたデータを取得して xml を生成する
public	updateData		QDomDocument	受信した xml を元に note.db を更新する
Public	init	void		テーブル、トリガの生成など初期化処理を行う
Public	databaseCreationScript	QString		Note アプリケーションで使用しているテーブルを構築するための CREATE 文を提供する
Public	LogCreationScript	QString		本ソフトウェアで追加するテーブルを構築するための CREATE 文を提供する

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
protected	m_database	QSqlDatabase		データベースへアクセスするためのクラス
private	itemList	QList<NoteItem>		更新されたデータを格納するクラス
private	receiveList	QList<NoteItem>		受信したデータを格納するクラス
private	splitter	QString	t_e*xt#	XML の Contents 属性に値を挿入する時、本文とタイトルなどを分割するための文字列

2.6 NoteItem クラス詳細

クラス詳細				
クラス名	NoteItem			
親クラス	なし			
可視性	Public			
クラス概要				
更新、受信したノートのデータを格納する				
インクルード				
1	QString			
メソッド				
可視性	メソッド名	型	引数	説明、その他
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	uuid	QString		データの uuid を格納する
public	content	QString		タイトル、本文などアプリケーションで使われているデータを格納する
public	lastUpdateTime	QString		最終更新日時を格納する
public	isValid	QString		データが削除されたかどうかのフラグ
public	pastSplit	int		前回のノートデータの分割数

				を格納する

2.7 SplittedItem クラス詳細

クラス詳細				
クラス名	SplittedItem			
親クラス	なし			
可視性	Public			
クラス概要				
ノートデータの XML 変換時に分割が行われたとき、分割されたデータの uuid と分割数を格納する				
インクルード				
1	QString			
メソッド				
可視性	メソッド名	型	引数	説明、その他
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	count	Int		分割が行われた回数
public	gid	QString		分割が行われたノートデータの UUID

2.8 Recent クラス詳細

クラス詳細			
クラス名	Recent		
親クラス	なし		
可視性	Public		
クラス概要			
履歴データが格納されているデータベース（chromium.db）に接続し、更新されたかどうかの監視とデータを受信した場合の更新処理を行う			
インクルード			
1	QObject	8	QVariant
2	QtSql/QSqlDatabase	9	QUuid
3	QtSql/QSqlQuery	10	QDomDocument
4	QtSql/QSqlRecord	11	QSettings
5	QSqlError	12	xmlpacket.h
6	QDir	13	recentItem.h
7	QSqlDriver	14	
メソッド			

可視性	メソッド名	型	引数	説明、その他
public	timerFunc	QString		データベースにアクセスし、変更されたデータを取得して xml を生成する
public	updateData		QDomDocument	受信した xml を元に chromium.db を更新する
Public	init	void		テーブル、トリガの生成など初期化処理を行う
Public	LogCreationScript	QString		本ソフトウェアで追加するテーブルを構築するための CREATE 文を提供する
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
protected	m_database	QSqlDatabase		データベースへアクセスするためのクラス
private	itemList	QList<RecentItem>		更新されたデータを格納するクラス
private	receiveList	QList<RecentItem>		受信したデータを格納するクラス
private	splitter	QString	t_e*xt#	XML の Contents 属性に値を挿入する時、URL とタイトルなどを分割するための文字列

2.9 RecentItem クラス詳細

クラス詳細				
クラス名	RecentItem			
親クラス	なし			
可視性	Public			
クラス概要				
更新、受信した履歴のデータを格納する				
インクルード				
1	QString			
メソッド				
可視性	メソッド名	型	引数	説明、その他
属性・インスタンス				

可視性	属性・インスタンス名	型	初期値	説明、その他
public	Uuid	QString		データの uuid を格納する
public	Content	QString		履歴表示の際に使われているデータを格納する
public	lastUpdateTime	QString		最終更新日時を格納する
public	isValid	QString		データが削除されたかどうかのフラグ

2.10 Calendar クラス詳細

クラス詳細				
クラス名	Calendar			
親クラス	なし			
可視性	Public			
クラス概要				
Calendar データが格納されているストレージから、更新されたかどうかの監視とデータを受信した場合の更新処理を行う				
インクルード				
1	QObject	10	QVariant	
2	QtSql/QSqlDatabase	11	QUuid	
3	QtSql/QSqlQuery	12	exception	
4	QtSql/QSqlRecord	13	xmlpacket.h	
5	QSqlError	14	calendarItem.h	
6	QDir	15	ekcal/ekcal-storage.h	
7	QSqlDriver	16	observer.h	
8	QSetting	17	calendardbsingleton.h	
9	QDomDocument			
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	timerFunc	QString		データベースにアクセスし、変更されたデータを取得して xml を生成する
public	updateData		QDomDocument	受信した xml を元にストレージを更新する
Public	init	void		テーブル、トリガの生成など初期化処理を行う
private	databaseName		QString	接続する SQLite のデータベース名を返す

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
protected	m_database	QSqlDatabase		データベースへアクセスするためのクラス
private	itemList	QList<NoteItem>		更新されたデータを格納するクラス
private	receiveList	QList<NoteItem>		受信したデータを格納するクラス
private	splitter	QString	t_e*xt#	XML の Contents 属性に値を挿入する時、日時とタイトルなどを分割するための文字列
private	syncCalendarSetting	QSettings		ini ファイルに最後に監視した時間を書きだすためのクラス
private	lastModifiedTime	QString		前回監視した時間を格納する

2.11 CalendarItem クラス詳細

クラス詳細				
クラス名	CalendarItem			
親クラス	なし			
可視性	Public			
クラス概要				
更新、受信した Calendar のデータを格納する				
インクルード				
1	QString			
メソッド				
可視性	メソッド名	型	引数	説明、その他
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	Uuid	QString		データの uuid を格納する
public	Content	QString		タイトル、日時などアプリケーションで使われているデータを格納する
public	lastUpdateTime	QString		最終更新日時を格納する
public	isValid	QString		データが削除されたかどうか

				のフラグ

2.12 CalendarDBSingleton クラス詳細

クラス詳細				
クラス名	CalendarDBSingleton			
親クラス	なし			
可視性	Public			
クラス概要				
Calendar データが格納されている Evolution Data Server にアクセスするためのインスタンスを生成する				
インクルード				
1	observer.h	2	ekcal/ekcal-storage.h	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	calendarPtr	Kcal::Calendar::Ptr		このクラスの calendar 変数を返す
public	sotragePtr	eKCal::@EStorage::Ptr		このクラスの storage 変数を返す
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	pinstance	CalendarDBSigleton		自身のインスタンスを格納する
private	calendar	KCal::Calendar::Ptr		Calendar にアクセスするためのインスタンス
private	storage	eKCal::EStorage::Ptr		Evolution Data Server にアクセスするためのインスタンス

2.13 Observer クラス詳細

クラス詳細			
クラス名	Observer		
親クラス	なし		
可視性	Public		
クラス概要			
Calendar アプリケーション、ストレージの変更を監視し、変更の通知を行う			
インクルード			
1	ekcal/ekcal-storage.h	6	QSqlDriver

2	icalformat.h	7	QSqlError
3	QtSql/QSqlDatabase	8	QDir
4	QtSql/QSqlQuery	9	QVaiant
5	QtSql/QSqlRecord	10	QObject

メソッド				
可視性	メソッド名	型	引数	説明、その他
public	beginIdleWatch		int	更新が一定時間行われたかどうかを監視するためのタイマーを作動させる
public	isIdle	bool		一定時間更新が行われたかどうかを返す
public	endIdleWatch			タイマーを停止する
protected	loadingComplete		bool,QString	ストレージのロードが完了した際に呼び出される
protected	savingComplete		bool,QString	ストレージのセーブが完了した際に呼び出される
protected	calendarInsidenceAdded		KCalCore::Insidence::Ptr	カレンダーにイベントが追加された時に呼び出される
protected	calendarInsidenceAdditionCanceled		KCalCore::Insidence::Ptr	イベントの追加が中止された時に呼び出される
protected	calendarInsidenceChanged		KCalCore::Insidence::Ptr	カレンダーのイベントが修正された時に呼び出される
protected	calendarInsidencedeleted		KCalCore::Insidence::Ptr	カレンダーのイベントが削除された時に呼び出される
protected	calendarModified		bool,KCalCore::Calendar	カレンダーが修正されたときに呼び出される
public	databaseName	QString		SQLite のデータベース名を返す

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
protected	m_database	QSqlDatabase		データベースへアクセスするためのクラス

2.14 Timethread クラス詳細

クラス詳細	
クラス名	TimeThread
親クラス	QThread

可視性	Public			
クラス概要				
比較処理、差分データを取る				
インクルード				
1	peoplemodel.h		7	
2	peoplemodel_p.h		8	
3	dbcopysave.h		9	
4	contact.h		10	
5			11	
6			12	
可視性	メソッド名	型	引数	説明、その他
public	readCotnactsFromlocal	QList		コンタクからデータを取る
Public	addContacts	void	QList	コンタクトにデータを追加
Public	removeContacts	void	QList	指定されたデータをコンタクトから削除する
Public	rewriteContacts	void	QList	指定されたデータをコンタクトから上書きする
Public	compareContacts	QList	QList,QList	コンタクトと CopyFile を比較
Public	compareContactsAfterReceive	QList	QList,QList	コンタクトと送られたデータを比較
Public	copyContactsToDBCopy	void	QList	差分データにより、DBCopyFile を更新する
Public	editContactsCopy	QList	QList,QList	送られたデータによって、Copy を更新
Public	readContactsFromFile	QList		CopyFile からデータを読み取る
Public	ContactsFileAddToFile	void	QList	CopyFile にデータを追加
Public	rewriteContactsToFile	void	QList	CopyFile にデータを上書き
Public	removeContactFromFile	void	QList	CopyFile にデータを削除
private	compareTime	int	QDateTime,QDateTime	時間を比較
メソッド・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	dbCopySave	DBCopy		インスタンス

		Save*		
private	contact	Contact*		インスタンス
public	contactsLocal	QList		
Public	contactsCopy	QList		
Public	contactsReceive	QList		
public	contactsSend	QList		
Public	contactsAdd	QList		
Public	contactRewrite	QList		
Public	ContactsRemove	QList		

2.15 XMLPacket クラス詳細

クラス詳細				
クラス名	XMLPacket			
親クラス	なし			
可視性	Public			
クラス概要				
XML 変換と解析				
インクルード				
1	QXmlStreamWriter		7	
2	QByterArray		8	
3	QDomDocument		9	
4	QDomElement		10	
5			11	
6			12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	commandParse	QString	QDomDocument	コマンドデータを XML 解析
public	contactParse	QList	QDomDocument	コンタクトデータを XML 解析
public	resultParse	QString	QDomDocument	解析の結果を取る
public	contactToXml	QString	QList	コンタクトデータを XML に変換
public	noteToXML	QStringList	QList	ノートデータを XML に変換
public	noteBookToXML	QStringList	QList	ノートブックデータを XML に変換
public	noteParse	QList	QDomDocument, QString	ノートデータ、ノートブックデータを XML 解析

public	recentToXML	QString	QList	履歴データを XML に変換
public	recentParse	QList	QDomDocument	履歴データを XML 解析
public	calendarToXML	QString	QList	カレンダーデータを XML に変換
public	calendarParse	QList	QDomDocument	カレンダーデータを XML 解析
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	splitSize	int	10000	ノートデータを分割する際の最大文字数を設定する

2.16 Config クラス詳細

クラス詳細				
クラス名	Config			
親クラス	なし			
可視性	Public			
クラス概要				
UI とネットワーク通信に関する設定				
インクルード				
1	QString		7	
2	QNetworkInterface		8	
3	QDebug		9	
4			10	
5			11	
6			12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	setHost	void	QStirng&	ホストの設計
Public	getHost	QString		ホストを取得
Public	setUser	void	QStirng&	UserId の設計
Public	getUser	QString		UserId を取得
Public	setPassword	void	QStirng&	パスワードの設計
Public	getPassword	QString		パスワードを取得
Public	setPort	void	Int	ポートの設計
Public	getPort	int		ポートを取得
Public	setDeviceId	void	QStirng&	デバイス Id の設計
Public	getDeviceId	QStirng&		デバイス Id を取得

Public	setConnectType	void	QStirng&	接続タイプの設計（ログイン、新規、パスワード変更）
Public	getConnectType	QStirng&		接続タイプを取得（ログイン、新規、パスワード変更）
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	m_host	QStirng		
private	m_port	Int		
private	m_user	QStirng		
private	m_password	QStirng		
private	m_deviceId	QStirng		
private	m_connectType	QStirng		

2.17 NetWork クラスの詳細

クラス詳細				
クラス名	NetWork			
親クラス	QObject			
可視性	Public			
クラス概要				
暗号化による socket 通信				
インクルード				
1	QObject		7	Config.h
2	QAbstractSocket		8	xmlPacket.h
3	QTcpSocket		9	Timethread.h
4	QBuffer		10	QRegExp
5	QDataStream		11	QhostAddress
6	QTime		12	QStringList
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	connectToServer	void	QString,int	サーバに接続
Public	discontactFromServer	void		サーバ側から切断
Public	isconnected	bool		切断状態のチェック
Public slots	handleStream	bool		接続したら、User とパスワードをサ

				ーバに送信
Public slots	socketReadRead	void		受信処理
Public slots	socketConnected	void		
Public slots	socketDisconnected	Void		
Public slots	login	void		
Public slots	receiveDataResult	void		
Public slots	sendContact	void	QString	コンタクトを送信
Public slots	ScanContact	void		コンタクトを取得
Public slots	sendMessage	void		Keep alive を送信
private	sendData	bool	QByteArray	文字列を送信
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	config	Config*		
Private	socket	QTCpSocket*		
Private	xmlPacket	XMLPacket*		
Private	timeThread	TimeThread*		
Private	message	QStirng		
Private	fileSize	Quint16		
Private	adduserCommandName	QStirng		
private	useLogonrCommandName	QStirng		
private	clientSendDataCommandName	QStirng		
private	serverSendDataCommandName	QStirng		

文書名

【ソフトウェア詳細設計書（サーバープログラム）】

文書番号

04-002

承認	作成
承認者名	劉宏超
承認日	2012/01/11

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考
1	2012/01/11	1.0		新規作成

目　次

1　概要

1.1　目的

本章はデータ共有システムサーバープログラム設計仕様について記述する

1.2　方針

1. 要求定義書と基本設計書に定めたことを基づいて、設計を行う。
2. マルティスレッドについて、.NetFramework の公開資料である MSDN をよく理解すること。

1.3　記載範囲

本章はサーバープログラムのソフトウェア構成、インタフェース、機能仕様を記載する

1.4　参照ドキュメント

ID	文書名	文書番号	発行年月	備考
	要件定義書			
	基本設計書			

1.5　定義（用語、略語）

ID	用語・略号	正式表記	意味
1	.NET	.NetFremework	サーバー側の開発環境である
2	MD5	Message Digest Algorithm 5	認証やデジタル署名などに使われるハッシュ関数である
3	AES	Advanced Encryption Standard	対称暗号化方式である
4	RSA		非対称暗号化方式である

2　クラス構造

2.1　詳細クラス図

インスタンス図と分析ラクスを基に、詳細クラス図を作成した。各クラスの詳細は別の表に説明する。

2.2 _ConnectionManager クラスの詳細

クラス詳細				
クラス名	**_ConnectionManager**			
親クラス	なし			
可視性	Public			
クラス概要				
デバイスの接続を管理するクラスである。				
インクルード				
1	System		7	System.Windows.Forms
2	System.Data		8	
3	System.Net.Sockets		9	
4	System.Collections.Generic		10	
5	System.Threading		11	
6	DBAccessor		12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	_ConnectionManager		DBConnStr PORT	コンストラクタメソッド
public	start	void		共有サービスを起動する
private	CreateManagerSocket	bool		サーバーソケットの作成
private	doManageConnection	void		接続をチェックし、セッションの作成、削除を行う
private	doCreateNewSession		clientSocket	新しいセッションの作成
private	doSendDataToSession	void	sender e	特定のデバイスにデータ送信
private	GetSessionIdFromDeviceId	int	devoceId	デバイス ID のセッション ID を取得
private	doProcessXMLStream	void		受信データを処理する
private	stop	bool		共有サービスを停止する
private	closeSocket	void	socket	サーバーソケットの削除
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	connectionManagerSocket	Socket		
private	DBConnStr	String		
private	PORT	Int		
private	sessionKey	Int	0	

private	deviceSessionList	Dictionary		
private	dataTable	DataTable		
private	m_SystemDBOperator	SystemDB Operator		
private	security	Security Transport Manager		
public	TBL	DataTable		モニタリング画面表示用

2.3 _Session ラスの詳細

クラス詳細	
クラス名	**_Session**
親クラス	なし
可視性	Public
クラス概要	
セッションクラス、デバイスとのデータ送受信などを管理するクラスである	
インクルード	

1	System	7	DBAccessor
2	System.Net	8	System.Windows.Forms
3	System.Net.Sockets	9	System.Threading
4	System.Text	10	
5	System.Collections.Generic	11	
6	System.IO	12	

メソッド

可視性	メソッド名	型	引数	説明、その他
public	_Session		serverSTM key deviceSocket m_SystemDB Operator	初期化
public	Close	void		終了
public	doProcessXMLStream	void		XML 処理指示
public	doBeginReceive	void		データ受信開始
private	GetPassedSeconds	double		前回受信から経過時間
private	SendDataCallBack	void	ira	送信後の CallBack
private	ReceiveDataCallBack	void	ira	受信後の CallBack
private	doProcessBuffer	void	readBytesLength securityTransport	バッファ処理を行う
public	OnSendDataToOtherSession	void	deviceId message	他のデバイスへ送信
public	SendToDevice	void	xmlString	デバイスへ送信

			securityTransport	
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	deviceId	string	空	デバイス ID
public	userName	string	空	ユーザ名
public	ip	string	空	IP アドレス
public	connectionStartTime	DateTime		接続した時間
private	lastSyncTime	DateTime		前回同期時間
private	packetSend	int	0	送信したパケット数
public	packetReceive	int	0	受信じたパケット数
public	KEY	int	0	
public	dataGridViewRow	DataGridViewRow		
public	security	SecurityTransportManager		
public	logon	bool	false	
public	userDBOperator	UserDBOperator		
private	systemDBOperator	SystemDBOperator		
public	sendDataToOtherSession	event EventHandler<SessionEventArgs>		
private	deviceSocket	Socket		
private	LastAccessTime	DateTime		
private	xmlStreamStart	string		
private	xmlStreamStop	string	ASCII 1	
private	xmlStreamBuffer	byte[]	ASCII 127	
private	xmlOperator	XMLOperator		

2.4 _XMLOperator クラス詳細

クラス詳細				
クラス名	**_XMLOperator**			
親クラス	なし			
可視性	Public			
クラス概要				
XML データの解析、処理を行うクラスである				
インクルード				
1	System	7	System.Data	
2	System.Text	8	OpenSSL.Core	
3	System.Collections.Generic	9		
4	System.IO	10		
5	DBAccessor	11		
6	System.Xml	12		
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	GetData	int	xmlDoc command data	XML ストリームからデータを取得する
public	InitContactTable	void		連絡先データー時的保存するテーブルの初期化
public	InitNoteTable	void		ノートデーター時的保存するテーブルの初期化
public	InitNoteBookTable	void		ノートブックデーター時的保存するテーブルの初期化
public	InitCalendarTable	void		カレンダーデーター時的保存するテーブルの初期化
public	InitHistoryTable	void		履歴データー時的保存するテーブルの初期化
public	doProcessXMLStream	void	session datagramBytes	XML データの処理を行う

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	contactTable	DataTable		
private	noteTable	DataTable		
private	noteBookTable	DataTable		
private	calendarTable	DataTable		
private	historyTable	DataTable		

2.5 _SecurityTransportManager クラス詳細

クラス詳細				
クラス名	_SecurityTransportManager			
親クラス	なし			
可視性	Public			
クラス概要				
セキュリティ関連、暗号化、復号化を行うクラスである				
インクルード				
1	System		7	System.Data
2	System.Text		8	OpenSSL.Core
3	System.Collections.Generic		9	
4	System.IO		10	
5	DBAccessor		11	
6	System.Xml		12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	InitRSA	void		コマンドデータを XML 解析
public	InitAES	void		コンタクトデータを XML 解析
public	SetAESInfo	void		解析の結果を取る
public	SetClientType	void		データを XML に変換
public	AESEncrypt	byte[]	val	AES 暗号化 外部アクセス用
public	AESDecrypt	byte[]	val	AES 復号化 外部アクセス用
private	GetRandomStr	string	lenth send	ランダム文字列を生成
private	GetMd5	string	pass	文字列の MD5 値を取得
private	ConvertHexStringToByteArray	byte[]	hexString	HEX 文字列の 元文字列を取得
private	GetClientType	ClientType	typeName	クライアントのタイプ（MeeGo デバイスかテストクライアント）を取得

private	AESEncrypt	byte[]	cc input key iv clientType	AES 暗号化を行う
private	AESDecrypt	byte[]	cc input key iv clientType	AES 復号化を行う
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
public	SecurityTransport	bool	false	
public	serverPublicKey	string	空	
public	serverPrivateKey	string	空	
public	clientTempStr	string	空	
private	myRSA	RSA		RSA インスタンス
private	myAES	AESSTRUCT		AES インスタンス
private	clientType	ClientType		

2.6 _ConfigReader クラス詳細

クラス詳細				
クラス名	**_ConfigReader**			
親クラス	なし			
可視性	Public			
クラス概要				
設定ファイルの読み込み専用クラスである				
インクルード				
1	System		7	
2	System.Collections.Generic		8	
3	QDebug		9	
4	System.Text		10	
5	System.IO		11	
6			12	
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	_ConfigReader		configFileName	コンストラクタメソッド
public	GetValue	int	DBIP PORT	パラメータ値を取得
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	configFileName	stirng	空	設定ファイルパス＋名前

2.7 _SessionEventArgs クラスの詳細

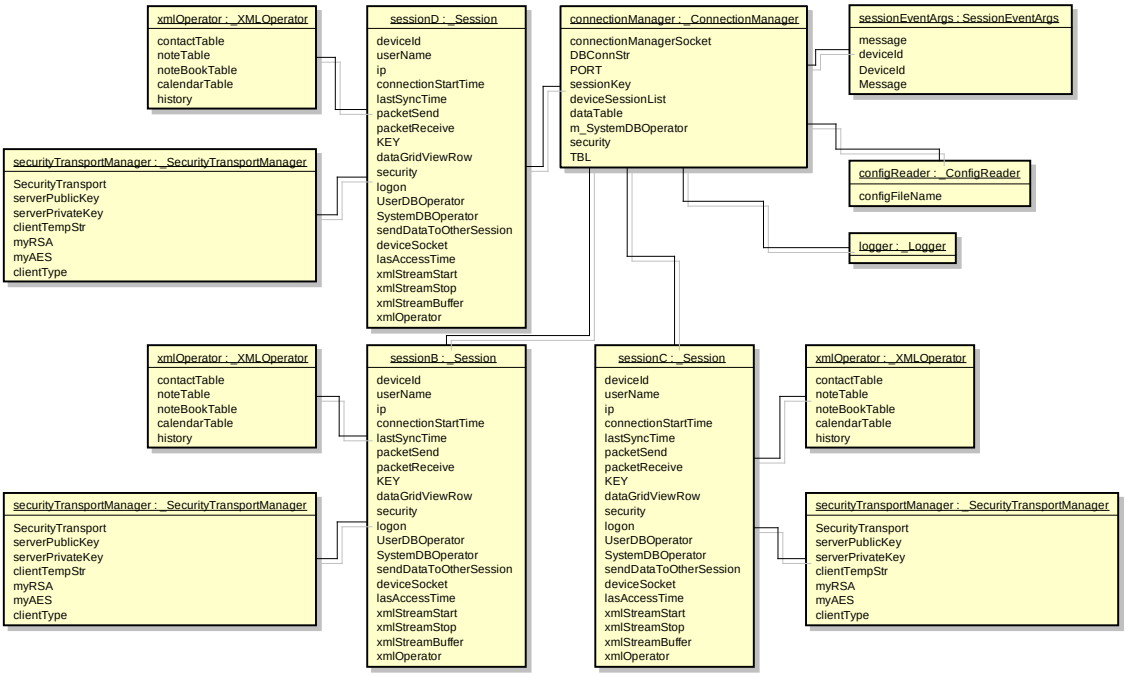
クラス詳細				
クラス名	**_SessionEventArgs**			
親クラス	EventArgs			
可視性	Public			
クラス概要				
他のでデバイスへ送信する用イベントクラスである				
インクルード				
1	System	7		
2	System.Collections.Generic	8		
3	System.Linq	9		
4	System.Text	10		
5		11		
6		12		
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	SessionEventArgs	void	deviceId message	サーバに接続
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	message	string	空	
Private	deviceId	string	空	
public	Message	string		message 取得専用
public	DeviceId	string		deviceId 取得専用

2.8 _Logger クラスの詳細

クラス詳細				
クラス名	**_Logger**			
親クラス	なし			
可視性	Public			
クラス概要				
ログ記録用クラスである				
インクルード				
1	System	7		
2	System.Collections.Generic	8		
3	System.Text	9		
4	System.IO	10		
5	System.Runtime.InteropServices	11		
6		12		
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	connectToServer	void	QString,int	サーバに接続
属性・インスタンス				

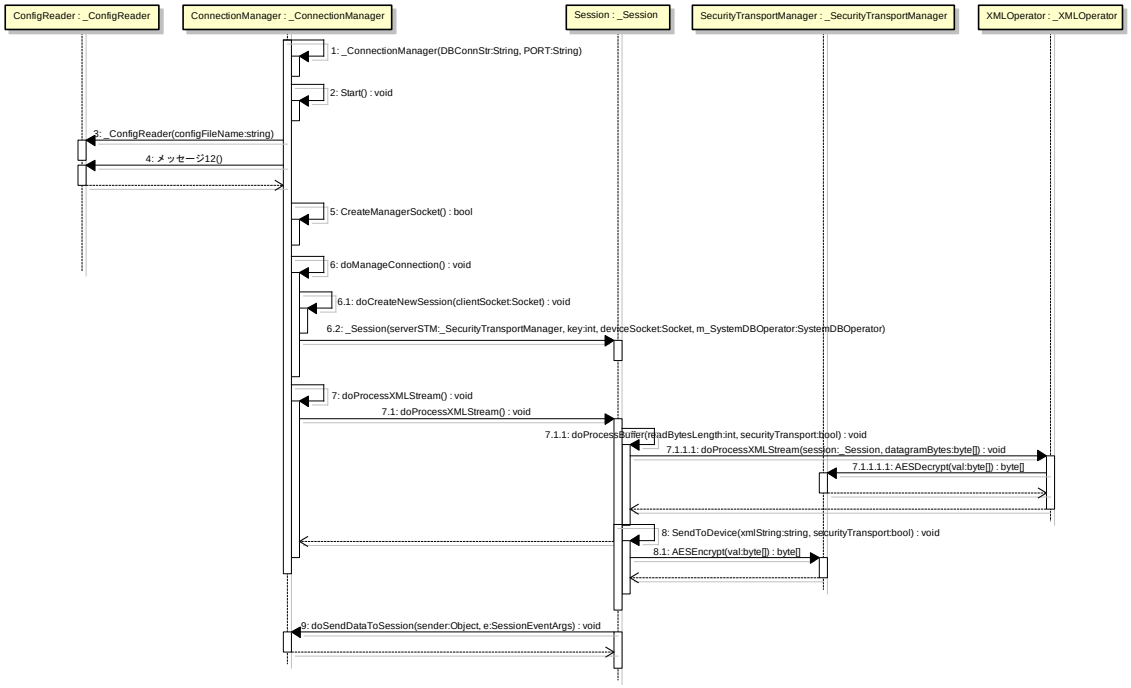
3　インスタンス図

詳細クラス図を基にサーバープログラムのシーケンス図を作成した。



4　シーケンス図

インスタンス図と要件定義を基にサーバープログラムのシーケンス図を作成した。



5　その他

- 設定ファイルの例は以下の通り。

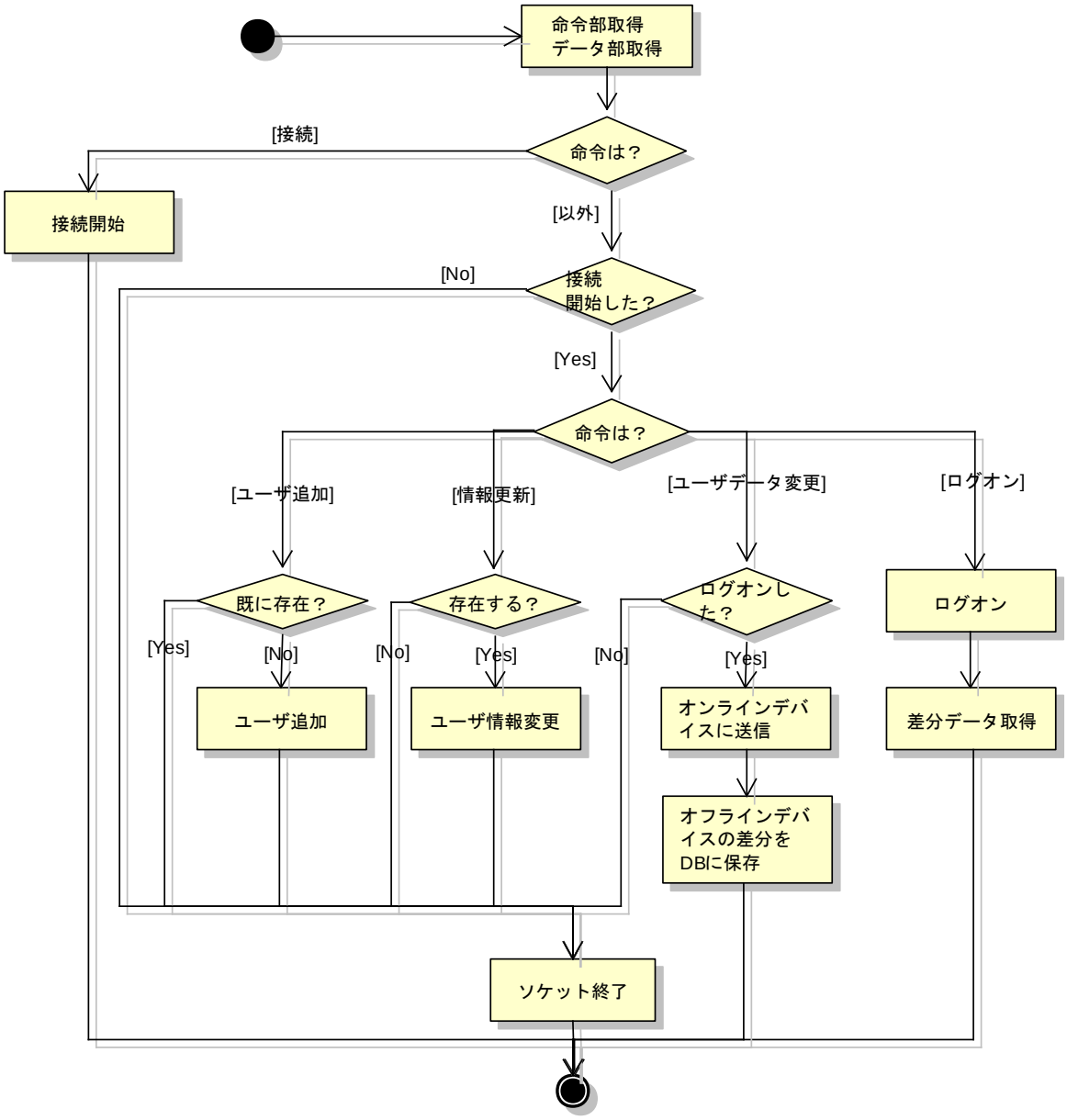
[DBConnStr]

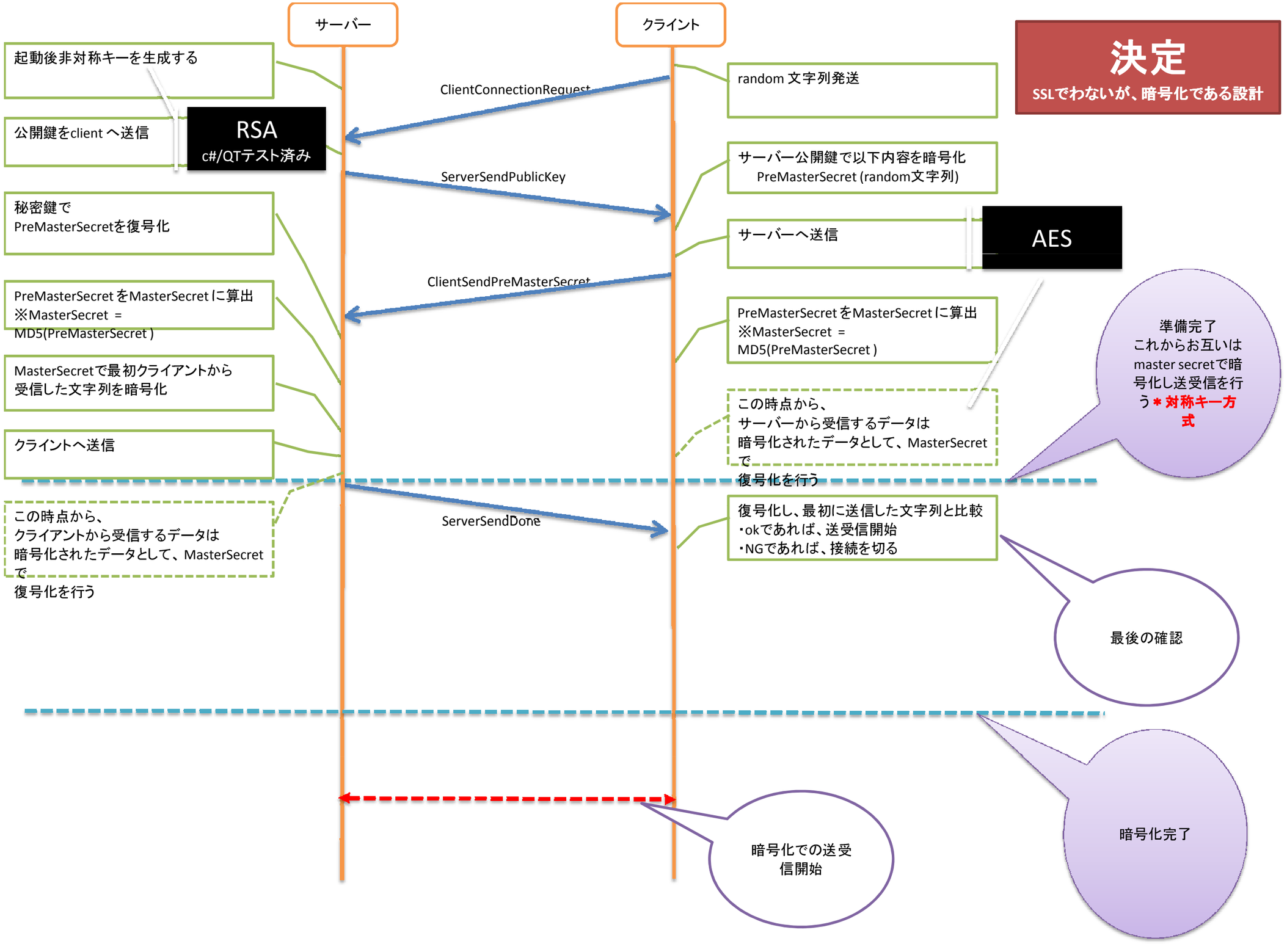
Server=[IP Address];UserId=[DBusername];Password=[DB pass];CharSet=utf8;Port=[port №];Max Pool Size=[xxx];

[Port]

xxxxx

- 本システム暗号化の詳細については資料「04-001-詳細設計書（暗号化).xlsx」を参考。
- XML 処理の流れな以下の通り。




サーバー
クライント
決定
SSLでわないが、暗号化である設計
起動後非対称キーを生成する
公開鍵をclient へ送信
RSA
c#/QTテスト済み
ClientConnectionReguest
random 文字列発送
ServerSendPublicKey
サーバー公開鍵で以下内容を暗号化
PreMasterSecret (random文字列)
秘密鍵で
PreMasterSecretを復号化
サーバーへ送信
AES
ClientSendPreMasterSecret
PreMasterSecret をMasterSecret に算出
※MasterSecret =
MD5(PreMasterSecret)
PreMasterSecret をMasterSecret に算出
※MasterSecret =
MD5(PreMasterSecret)
準備完了
これからお互いは
master secretで暗
号化し送受信を行
う＊対称キー方
式
MasterSecretで最初クライアントから
受信した文字列を暗号化
この時点から、
サーバーから受信するデータは
暗号化されたデータとして、MasterSecret
で
復号化を行う
クライントへ送信
この時点から、
クライアントから受信するデータは
暗号化されたデータとして、MasterSecret
で
復号化を行う
ServerSendDone
復号化し、最初に送信した文字列と比較
・okであれば、送受信開始
・NGであれば、接続を切る
最後の確認
暗号化完了
暗号化での送受
信開始

文書名

【ソフトウェア詳細設計書（データベース）】

文書番号

04-004

承認	作成
承認者名	劉建宇
承認日	作成日

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考

目　次

1　概要

1.1　目的

本書はデータ共有システムクライント側のソフトウェア設計仕様について記述する

1.2　方針

初めて MeeGo デバイスをベースにしたソフトウェア開発であるため、下記の点に重点を置きった。

1. 要求定義書と基本設計書に定めたことを基づいて、設計を行う。
2. MeeGo はオーペンソースなので、そのドキュメントやサンプルプログラムをより理解し、参考にする。

1.3　記載範囲

本書はクライント側のソフトウェア構成、インタフェース、機能仕様を記載する

1.4　参照ドキュメント

ID	文書名	文書番号	発行年月	備考
	要件定義書			
	基本設計書			

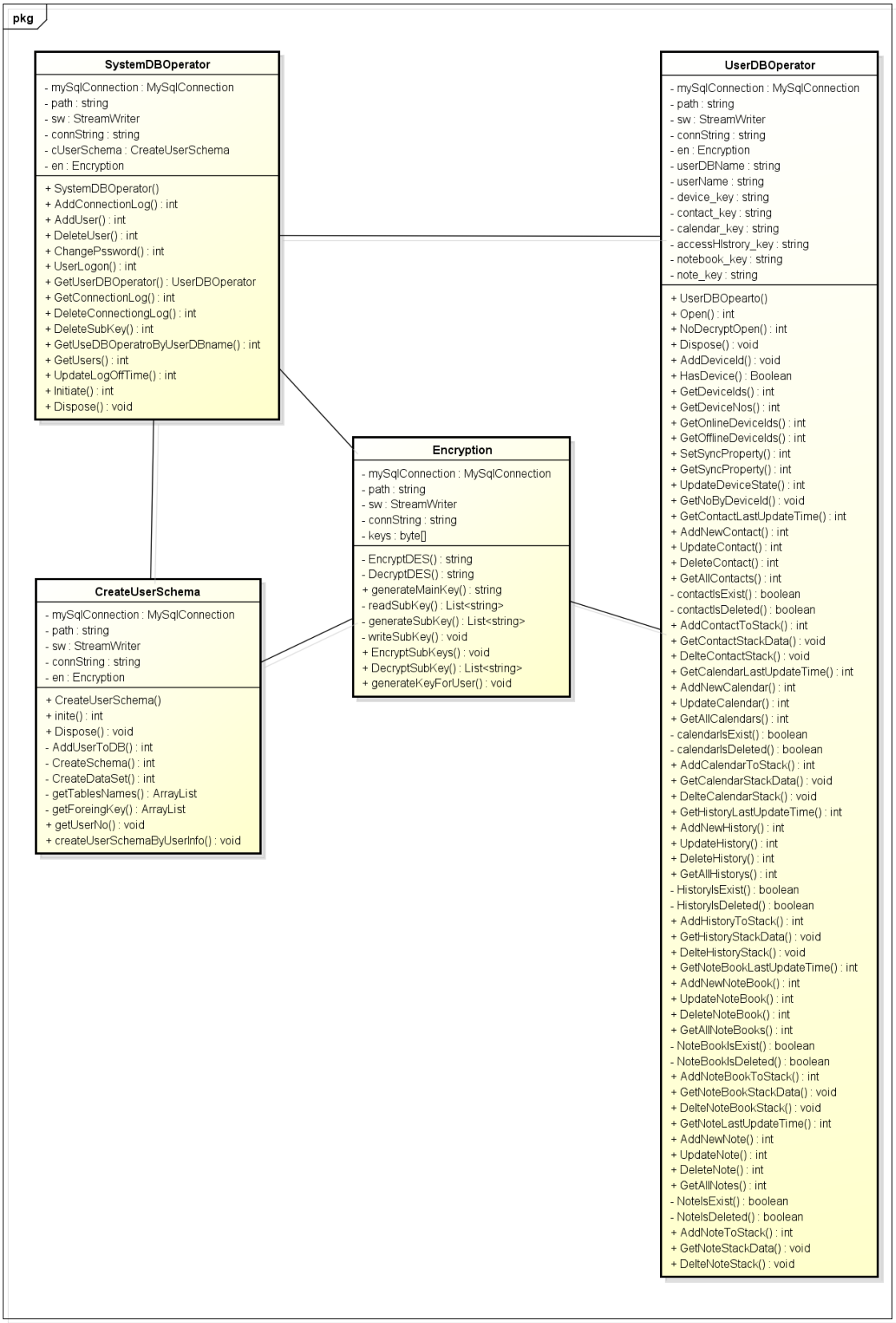
1.5　定義（用語、略語）

ID	用語・略号	正式表記	意味

2　クラス構造

2.1　設計クラス図

分析ラクスを基に、設計クラス図を作成した。各クラスの詳細は別の表に説明する。


pkg
SystemDBOperator
- mySqlConnection : MySqlConnection
- path : string
- sw : StreamWriter
- connString : string
- cUserSchema : CreateUserSchema
- en : Encryption
+ SystemDBOperator()
+ AddConnectionLog() : int
+ AddUser() : int
+ DeleteUser() : int
+ ChangePssword() : int
+ UserLogon() : int
+ GetUserDBOperator() : UserDBOperator
+ GetConnectionLog() : int
+ DeleteConnectiongLog() : int
+ DeleteSubKey() : int
+ GetUseDBOperatroByUserDBname() : int
+ GetUsers() : int
+ UpdateLogOffTime() : int
+ Initiate() : int
+ Dispose() : void
UserDBOperator
- mySqlConnection : MySqlConnection
- path : string
- sw : StreamWriter
- connString : string
- en : Encryption
- userDBName : string
- userName : string
- device_key : string
- contact_key : string
- calendar_key : string
- accessHistrory_key : string
- notebook_key : string
- note_key : string
+ UserDBOpearto()
+ Open() : int
+ NoDecryptOpen() : int
+ Dispose() : void
+ AddDeviceId() : void
+ HasDevice() : Boolean
+ GetDeviceIds() : int
+ GetDeviceNos() : int
+ GetOnlineDeviceIds() : int
+ GetOfflineDeviceIds() : int
+ SetSyncProperty() : int
+ GetSyncProperty() : int
+ UpdateDeviceState() : int
+ GetNoByDeviceId() : void
+ GetContactLastUpdateTime() : int
+ AddNewContact() : int
+ UpdateContact() : int
+ DeleteContact() : int
+ GetAllContacts() : int
- contactIsExist() : boolean
- contactIsDeleted() : boolean
+ AddContactToStack() : int
+ GetContactStackData() : void
+ DelteContactStack() : void
+ GetCalendarLastUpdateTime() : int
+ AddNewCalendar() : int
+ UpdateCalendar() : int
+ GetAllCalendars() : int
- calendarIsExist() : boolean
- calendarIsDeleted() : boolean
+ AddCalendarToStack() : int
+ GetCalendarStackData() : void
+ DelteCalendarStack() : void
+ GetHistoryLastUpdateTime() : int
+ AddNewHistory() : int
+ UpdateHistory() : int
+ DeleteHistory() : int
+ GetAllHistorys() : int
- HistoryIsExist() : boolean
- HistoryIsDeleted() : boolean
+ AddHistoryToStack() : int
+ GetHistoryStackData() : void
+ DelteHistoryStack() : void
+ GetNoteBookLastUpdateTime() : int
+ AddNewNoteBook() : int
+ UpdateNoteBook() : int
+ DeleteNoteBook() : int
+ GetAllNoteBooks() : int
- NoteBookIsExist() : boolean
- NoteBookIsDeleted() : boolean
+ AddNoteBookToStack() : int
+ GetNoteBookStackData() : void
+ DelteNoteBookStack() : void
+ GetNoteLastUpdateTime() : int
+ AddNewNote() : int
+ UpdateNote() : int
+ DeleteNote() : int
+ GetAllNotes() : int
- NoteIsExist() : boolean
- NoteIsDeleted() : boolean
+ AddNoteToStack() : int
+ GetNoteStackData() : void
+ DelteNoteStack() : void
Encryption
- mySqlConnection : MySqlConnection
- path : string
- sw : StreamWriter
- connString : string
- keys : byte[]
- EncryptDES() : string
- DecryptDES() : string
+ generateMainKey() : string
- readSubKey() : List<string>
- generateSubKey() : List<string>
- writeSubKey() : void
+ EncryptSubKeys() : void
+ DecryptSubKey() : List<string>
+ generateKeyForUser() : void
CreateUserSchema
- mySqlConnection : MySqlConnection
- path : string
- sw : StreamWriter
- connString : string
- en : Encryption
+ CreateUserSchema()
+ inite() : int
+ Dispose() : void
- AddUserToDB() : int
- CreateSchema() : int
- CreateDataSet() : int
- getTablesNames() : ArrayList
- getForeingKey() : ArrayList
+ getUserNo() : void
+ createUserSchemaByUserInfo() : void

2.2　利用したライブラリ

インクルード			
1	System	8	System.Data.SqlClient
2	System.IO	9	System.Web
3	System.Collections.Generic	10	MySql.Data.MySqlClient
4	System.ComponentModel	11	System.Collections
5	System.Data	12	System.Security.Cryptography
6	System.Text	13	System.Web.Security
7	System.Diagnostics	14	System.Threading

2.3　SystemDBOperator クラスの詳細

クラス詳細				
クラス名	SystemDBOperator			
親クラス	なし			
可視性	Public			
クラス概要				
システムスキーマ操作モジュール				
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	SystemDBOperator			構造関数
public	Initiate	int	string	システムスキーマ操作モジュールの初期化
public	Dispose	void		データベースへの連結を切る
public	AddUser	int	string	新規ユーザーを登録する
public	DeleteUser	int	string	指定されたユーザーを削除する
public	ChangePassword	int	string	指定されたユーザーのパスワードを修正する
public	UserLogon	int	string	ユーザー名とパスワードを利用し、ユーザーを検証する
public	UserDBOperator	GetUserDBOperator	string	ユーザー名を利用し、ユーザースキーマ操

				作のハンドルを獲得する
public	GetUsers	Int	list<string>	データベースに存在しているすべてのユーザーナンバーを返す
public	AddConnectionLog	int	string, DateTime	ユーザーの接続履歴をデータベースに追加
public	GetConnectionLog	int	string, DateTime	ユーザー名とデバイス名を利用し、接続履歴を獲得する
public	DeleteConnectionLog	int	string	接続履歴を削除する
public	UpdateLogOffTime	int	string, DateTime	ログオフ時間を追加する
属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	mySqlConnection	MySqlConnection *		
Private	path	string		
Private	sw	StreamWriter		
Private	connString	string		
Private	en	Encryption		
Private	CreateUserSchema	cUserSchema		

2.4 UserDBOperator ラスの詳細

クラス詳細				
クラス名	UserDBOperator			
親クラス	なし			
可視性	Public			
クラス概要				
ユーザースキーマを操作するためのクラス				
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	UserDBOperator			ユーザースキーマの構造関数
public	Initiate	int		ユーザースキーマ操作モジュールの初期化
Public	Dispose	void		データベースへの連結を

				切る
Public	AddDeviceId	int	string	新規デバイスを追加する
Public	HasDevice	boolean	string	デバイスが存在するかどうかを判断する
Public	GetDeviceIds	int	List<string>	ユーザーに属するすべてのデバイス ID を獲得する
Public	GetDeviceNos	Int	List<int>	ユーザーに属するすべてのデバイスナンバーを獲得する
Public	DBCopySave	int	List<string>	オンラインデバイス ID を獲得する
public	GetOfflineDeviceIds	int	List<string>	オフラインデバイス ID を獲得する
public	SetSyncProperty	Int	String Bool	ユーザーデータの同期するかどうかを設定する
public	GetSyncProperty	Int	String Bool	同期設定を獲得する
public	UpdateDeviceState	Int	string	デバイスのオンライン状態を更新する
public	GetNoByDeviceId	Int	String	デバイスナンバーを獲得する
public	GetContactLastUpdateTime	int	String Datetime	あるコンタクト前回更新した時間を取得する
public	AddNewContact	int	String Datetime	コンタクトを追加する
public	UpdateContact	int	String Datetime	コンタクトを更新する
public	DeleteContact	int	String	コンタクトを無効する
public	GetAllContacts	int	datatable	新ディバイス対応ユーザコンタクトを全部取得
private	contactIsExist	Boolean	string	コンタクトが存在かどうかを判断する
private	contactIsDeleted	Boolean	string	コンタクトが無効かどうかを判断する
public	AddContactToStack	int	string	オフラインデバイスの更新すべき contactid を追加する

public	GetContactStackData	int	String datatable	デバイスが stack データを取得
public	DeleteContactStack	int	string	デバイスにで GetContactStackData 取ったデータを発送成功した時点呼び出す
public	AddNewCalendar	int	String Datetime	カレンダイベントーを追加する
public	UpdateCalendar	int	String Datetime	カレンダイベントーを更新する
public	DeleteCalendar	int	String	カレンダイベントーを無効する
public	GetAllCalendars	int	datatable	新ディバイス対応ユーザカレンダイベントーを全部取得
private	calendarIsExist	Boolean	string	カレンダイベントーが存在かどうかを判断する
private	calendarIsDeleted	Boolean	string	カレンダイベントーが無効かどうかを判断する
public	AddCalendarToStack	int	string	オフラインデバイスの更新すべき calendarid を追加する
public	GetCalendarStackData	int	String datatable	デバイスが stack データを取得
public	DeleteCalendarStack	int	string	デバイスにで GetCalendarStackData 取ったデータを発送成功した時点呼び出す
public	AddNewHistory	int	String Datetime	閲覧履歴を追加する
public	UpdateHistory	int	String Datetime	閲覧履歴を更新する
public	DeleteHistory	int	String	閲覧履歴を無効する
public	GetAllHistorys	int	datatable	新ディバイス対応ユーザ閲覧履歴を全部取得
private	HistoryIsExist	Boolean	string	閲覧履歴が存在かどうかを判断する

private	HistoryIsDeleted	Boolean	string	閲覧履歴が無効かどうかを判断する
public	AddHistoryToStack	int	string	オフラインデバイスの更新すべき Historyid を追加する
public	GetHistoryStackData	int	String datatable	デバイスが stack データを取得
public	DeleteHistoryStack	int	string	デバイスにで GetHistoryStackData 取ったデータを発送成功した時点呼び出す
public	AddNewNoteBook	int	String Datetime	ノートブックを追加する
public	UpdateNoteBook	int	String Datetime	ノートブックを更新する
public	DeleteNoteBook	int	String	ノートブックを無効する
public	GetAllNoteBooks	int	datatable	新ディバイス対応ユーザノートブックを全部取得
private	NoteBookIsExist	Boolean	string	ノートブックが存在かどうかを判断する
private	NoteBookIsDeleted	Boolean	string	ノートブックが無効かどうかを判断する
public	AddNoteBookToStack	int	string	オフラインデバイスの更新すべき NoteBookid を追加する
public	GetNoteBookStackData	int	String datatable	デバイスが stack データを取得
public	DeleteNoteBookStack	int	string	デバイスにで GetNoteBookStackData 取ったデータを発送成功した時点呼び出す
public	AddNewNote	int	String Datetime	ノートを追加する
public	UpdateNote	int	String Datetime	ノートを更新する
public	DeleteNote	int	String	ノートを無効する
public	GetAllNotes	int	datatable	新ディバイス対応ユーザ

				ノートを全部取得
private	NoteIsExist	Boolean	string	ノートが存在かどうかを判断する
private	NoteIsDeleted	Boolean	string	ノートが無効かどうかを判断する
public	AddNoteToStack	int	string	オフラインデバイスの更新すべき Noteid を追加する
public	GetNoteStackData	int	String datatable	デバイスが stack データを取得
public	DeleteNoteStack	int	string	デバイスにで GetNoteStackData 取ったデータを発送成功した時点呼び出す

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	mySqlConnection	MySqlConnection *		
Private	path	string		
Private	sw	StreamWriter		
Private	connString	string		
Private	en	Encryption		
Private	CreateUserSchema	cUserSchema		

2.5 CreateUserSchma クラス詳細

クラス詳細	
クラス名	**CreateUserSchma**
親クラス	なし
可視性	Public
クラス概要	
新規ユーザー登録するとき、ユーザーのスキーマを作成するためのクラス	

メソッド				
可視性	メソッド名	型	引数	説明、その他
public	CreateUserSchema			クラスの構造関数
public	inite	void		初期化関数
public	Dispose	void		データベースへの連結を切る
public	createUserSchemaByUserInfo	int	string	ユーザー情報を利用し、ユーザースキーマを作成する
private	AddUserToDB	void	string	新規ユーザーの情報をシステ

				ムスキーマに追加する
private	CreateSchema	void	string	ユーザースキーマを作成する
private	CreateDataSet	void	string	ユーザーに属するテーブルを作成する
private	getTablesNames	ArrayList		sampleDB のテーブル名を獲得する
private	getForeignKey	ArrayList		sampleDB の外部キー制約を獲得する
public	getUserNo	int	string	ユーザー名を利用し、ユーザーのナンバーを獲得する

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	mySqlConnection	MySqlConnection *		
Private	path	string		
Private	sw	StreamWriter		
Private	connString	string		
Private	en	Encryption		

2.6 Encryption クラス詳細

クラス詳細	
クラス名	Encryption
親クラス	なし
可視性	Public
クラス概要	
暗号化するためのクラス	

メソッド				
可視性	メソッド名	型	引数	説明、その他
public	Encryption		string	構造関数
private	EncryptDES	string	string	DES 暗号化関数
private	DecryptDES	string	string	DES 復号化関数
private	readSubKey	List<string>	string	データベースからサブキーを獲得する
public	generateMainKey	string	string	生成メインキー
private	generateSubKey	List<string>	int	生成サブキー
private	writeSubeKey	void	List<string>、string	生成したサブキーをデータベースに保存する
private	EncryptsubKeys	void	List<string>、	メインキーを利用し、サブ

			string	キーを暗号化する。
public	DecryptSubKey	List<string>	string	メインキーを利用し、サブキーを復号化する
public	generateKeyForUser	void	string	ユーザーのために、メインキーサブキーを生成する

属性・インスタンス				
可視性	属性・インスタンス名	型	初期値	説明、その他
private	mySqlConnection	MySqlConnection *		
Private	path	string		
Private	sw	StreamWriter		
Private	connString	string		
Private	Keys	byte[]	0x14, 0x32, 0x58, 0x71, 0x95, 0xAE, 0xED, 0xEC	

文書名

【ソフトウェア詳細設計書（XML 仕様）】

文書番号

04-005

	作成
作成者名	劉　建宇
修正者名	劉　宏超 許　先華
作成日	2011 年 8 月 10 日

発行日

発行部署

改訂履歴

項番	日付	バージョン	改訂内容	備考
1	2012/12 /10	V1.1	同期処理	
1	2012/12 /10	V1.2	セキュリティ関係	

目　次

1 概要

1.1 目的

本書はデータ共有システムのクライアントとサーバー通信を行う XML データ仕様について記述する

1.2 方針

初めて MeeGo デバイスをベースにしたソフトウェア開発であるため、下記の点に重点を置きった。

1. 要求定義書と基本設計書に定めたことを基づいて、設計を行う。
2. XML 標準仕様を参考にして設計を行う。

1.3 記載範囲

本書はクライント側のソフトウェア構成、インタフェース、機能仕様を記載する

1.4 参照ドキュメント

ID	文書名	文書番号	発行年月	備考
	要件定義書			
	基本設計書			

1.5 定義（用語、略語）

ID	用語・略号	正式表記	意味
	XML		

2 データ通信関係

2.1 接続準備（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>BeginConnection</Command>
 <Data>
  <DeviceId>device1</DeviceId>
 </Data>
</MeegoSync>
```

2.2　接続準備返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>BeginConnection</Command>
 <Data></Data>
 <Result>
  <Code></Code> <0:ok else:error>
  <Message></Message><!
 </Result>
</MeegoSync>
```

2.3　ログイン（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>UserLogon</Command>
 <Data>
        <Certification>
                <UserName>qqq,qqq</UserName>
                <Password></Password>
        </Certification>
        <SyncOption>
                <Contact></Contact>
                <Calendar></Calendar>
                <History></History>
        </SyncOption>
 </Data>
</MeegoSync>
```

2.4　ログイン返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>UserLogon</Command>
 <Data>
        <Contact>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
        </Contact>
```

```
        <Contact>
            <Uuid></Uuid>
            <Content></Content>
            <LastUpdateTime></LastUpdateTime>
        </Contact>
        <Calendar></Calendar>
        <History></History>
  </Data>
  <Result>
    <Code></Code>
    <Message></Message>
  </Result>
</MeegoSync>
```

2.5　ユーザー登録（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
  <Command>AddUser</Command>
  <Data>
        <Certification>
            <UserName></UserName>
            <Password></Password>
        </Certification>
  </Data>
</MeegoSync>
```

2.6　ユーザー登録返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
  <Command>AddUser</Command>
  <Data>
  </Data>
  <Result>
    <Code></Code><!--0:f[^MOK else:errorcode>
    <Message></Message>
  </Result>
</MeegoSync>
```

2.7　ユーザー情報変更（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>UserInfoUpdate</Command>
 <Data>
        <Certification>
                <UserName></UserName>
                <Password></Password>
        </Certification>
        <ChangeInfo>
                <Password></Password>
        </ChangeInfo>
 </Data>
</MeegoSync>
```

2.8　ユーザー情報変更返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>UserInfoUpdate</Command>
 <Data>
 </Data>
 <Result>
  <Code><Code><!--0:f[^MOK else:errorcode>
  <Message><Message><!--e>
 </Result>
</MeegoSync>
```

2.9　コネクション保持（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>KeepAlive</Command>
</MeegoSync>
```

2.10　コネクション保持返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>KeepAlive</Command>
 <Result>
```

```
  <Code><Code><!--0:OK>
 </Result>
</MeegoSync>
```

2.11 同期処理（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ClientSendData</Command>
 <Data>
        <Contact>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>0</IsValid> //データが削除する場合は「1」
        </Contact>
        <Note>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>0</IsValid> //データが削除する場合は「1」
        </Note>
        <NoteBook>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>1</IsValid> //データが削除する場合は「1」
        </NoteBook>
        <Calendar></Calendar>
        <History></History>
 </Data>
</MeegoSync>
```

2.12 同期処理返信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ClientSendData</Command>
 <Data>
```

```
 </Data>
 <Result>
  <Code><Code><!--0:f[^MOK else:errorcode>
  <Message><Message><!--燉 e>
 </Result>
</MeegoSync>
```

2.13 同期処理（サーバーからクライアント）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ServerSendData</Command>
 <Data>
        <Contact>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>0</IsValid> //データが削除され、無効な場合は「1」

        </Contact>
        <Contact>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>0</IsValid> //データが削除され、無効な場合は「1」
        </Contact>
        <Contact>
                <Uuid></Uuid>
                <Content></Content>
                <LastUpdateTime></LastUpdateTime>
                <IsValid>0</IsValid> //データが削除され、無効な場合は「1」
        </Contact>
        <Calendar></Calendar>
        <History></History>
 </Data>
</MeegoSync>
```

説明1:
データが有効な場合には「0」を返し、データが無効な場合には「1 」を返します

2.14 同期処理返信（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ServerSendData</Command>
 <Data>
 </Data>
 <Result>
  <Code><Code><!--0:データ送信 OK else:errorcode>
  <Message><Message><!--内容>
 </Result>
</MeegoSync>
```

3 セキュリティ信関係

3.1 接続要求（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ClientConnectionRequest</Command>
 <Data>
  <ClientTempStr>XXXXXXXXXX</ClientTempStr>
  <ClientType>QT[CSHARP]</ClientType> [QT では省略可-初期値のため]
 </Data>
</MeegoSync>
```

3.2 サーバー公開鍵送信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ServerSendPublicKey</Command>
 <Data>
  <PublicKey>XXXXXXXXXX</PublicKey>
 </Data>
</MeegoSync>
```

3.3 PreMasterSecret 送信（クライアントからサーバーへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ClientSendPreMasterSecret</Command>
```

```
 <Data>
  <PreMasterSecret>XXXXXXXXXX</PreMasterSecret>
 </Data>
</MeegoSync>
```

3.4 SendDone 送信（サーバーからクライアントへ）

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ServerSendDone</Command>
 <Data>
  <ClientTempStr>XXXXXXXXXX</ClientTempStr>
 </Data>
</MeegoSync>
```

データベース設計書

第 1 版

筑波大学大学院
システム情報工学研究科コンピュータサイエンス専攻
高度 IT 人材育成のためのソフトウェア開発専修プログラム

チーム

作成年月日　平成 23 年 9 月 11 日

変更履歴

変更日	変更内容
2011/9/11	第 1 版作成
2011/9/12	ユーザーID は整数から文字列に変更、ユーザーname と電話を削除する。
2011/9/15	

目次

1. データベース設計書について

1.1　本書の目的

本設計書は、meego データ共有システムにおける論理データモデル、物理データモデルを定義し、その内容について我々開発チームの認識を一致させることを目的とする。

1.2　本書の想定読者

IT システムについての基本的な知識を有し、meego データ共有システムのメンバーを想定している。

1.3　本書の構成

本書は 2 つの章から構成される。

第 1 章では、本書の目的、想定読者、構成について示す。

第 2 章では、開発するシステムのデータモデル定義を示す。

2. データモデル定義

2.1 名称付与基準

2.1.1 目的・方針

データの重複に起因した業務の不整合をなくすため、meego データ共有システムシステム内のあらゆるデータ項目名称の付与基準を定める。

2.1.2 付与対象

Meego データ共有システムの全てのエンティティとその属性を対象とする。

2.1.3 体制・役割

名称付与については、データ管理者が一元的に管理し、名称の追加、変更、削除を行う。

2.1.4 名称付与基準

エンティティの名称は、標準用語定義に登録された用語、もしくはそれらを組み合わせたものでなければならない。

2.1.5 用語作成手順

(1) 所定の用語が標準用語 DB に登録されているか検索する。

(2) 新しい用語の登録が必要な場合は、名称付与の依頼をデータ管理者に行う。

(3) データ管理者は登録の必要性をチェックし、登録が必要な用語を標準用語 DB に登録する。

2.1.6 名称付与方法

標準用語定義を用いて名称を作成し、ユニークな ID、英語名称等を付与する。

2.2 標準用語定義

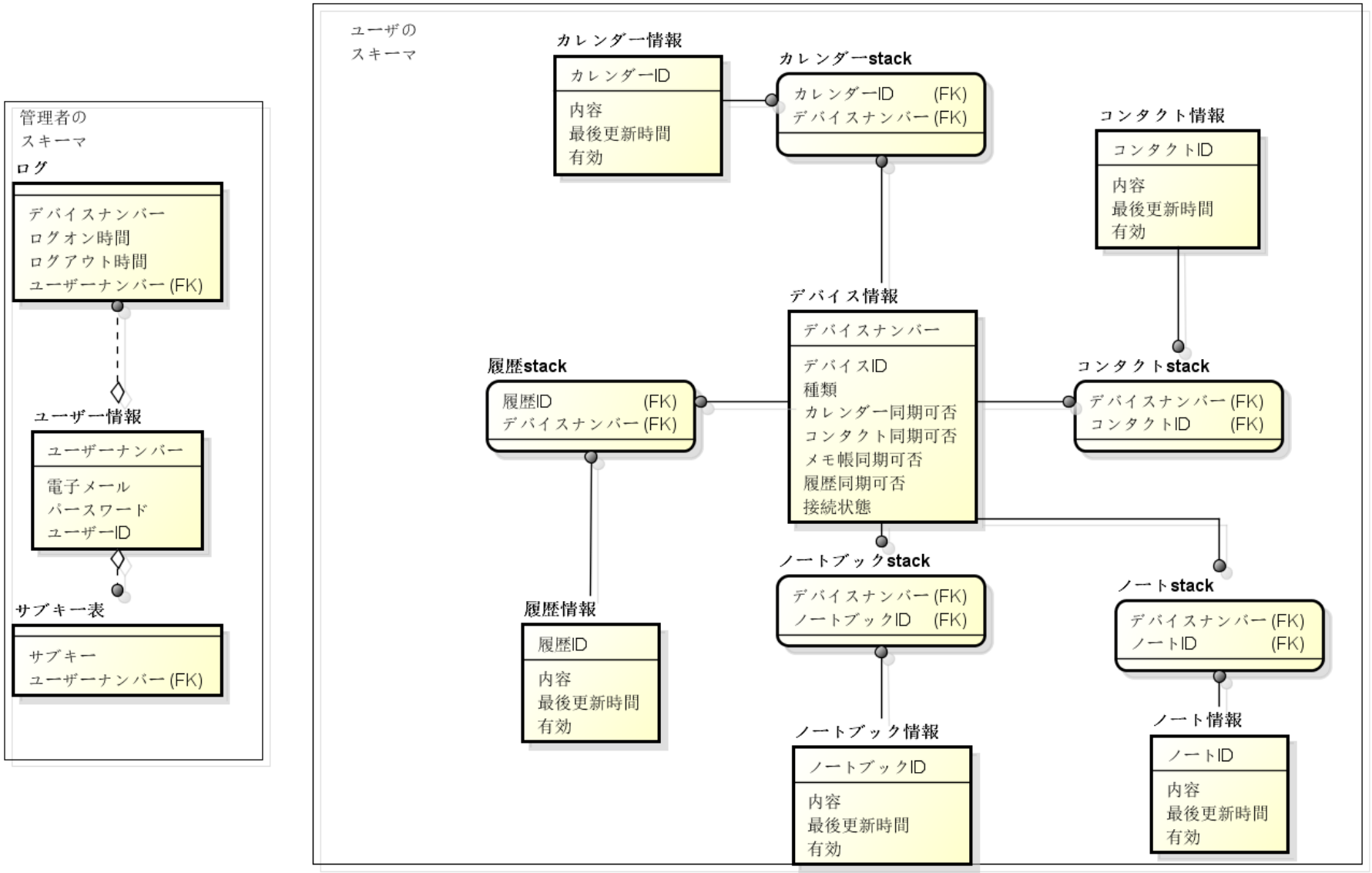
ID	標準用語	フリガナ	物理名	略語	標準用語説明
D001	ユーザー	ユーザー	USER		Meego データ共有システムサービスに登録したユーザー
D002	電子メール	デンシメール	E-MAIL		ユーザーの電子メールアドレス
D003	パースワード	パースワード	PASSWORD		
D004	デバイス	デバイス	DEVICE		ユーザーを持っている meego デバイス
D006	ログイン	ログイン	LOGIN		
D007	ログアウト	ログアウト	LOGOUT		
D008	ID	アイディ	ID		DB 上のテーブルで一意に管理するための値
D09	時間	ジカン	TIME		
D010	種類	シュルイ	TYPE		タブレット、ネットブック、ハンドサイトなど
D011	カレンダー	カレンダー	CALENDAR		
D012	コンタクト	コンタクト	CONTACT		
D013	履歴	リレキ	HISTORY		メディアファイル、book などの閲覧履歴
D014	接続状態	セツゾクジョウタイ	CONNECTIONSTATUS		デバイスの接続の状態
D015	内容	ナイヨウ	CONTENT		ユーザーのカレンダー、コンタクト、履歴などのデータ
D016	最後更新	サイゴコウシン	LASTUPDATE		
D017	有効	ユウコウ	VALID		
D018	完成	カンセイ	FINISH		
D019	同期	ドウキ	SYNCHRONIZATION		ユーザーデバイスとデバイス、デバイスとサーバ間の同期
D020	筆記帳	ノートブック	NOTEBOOK		

ID	標準用語	フリガナ	物理名	略語	標準用語説明
D021	筆記	ノート	NOTE		
D022	ログ	ログ	LOG		
D023	スタック	スタック	STACK		デバイスがオフライン時、同期データの格納場所
D024	情報	ジョウホウ	INFORMATION		
D025	可否	カヒ	IS		
D026	サブキー	サブキー	SUBKEY		データベースのデータを暗号化するためのキー
D027	ナンバー	ナンバー	NUMBER	No	

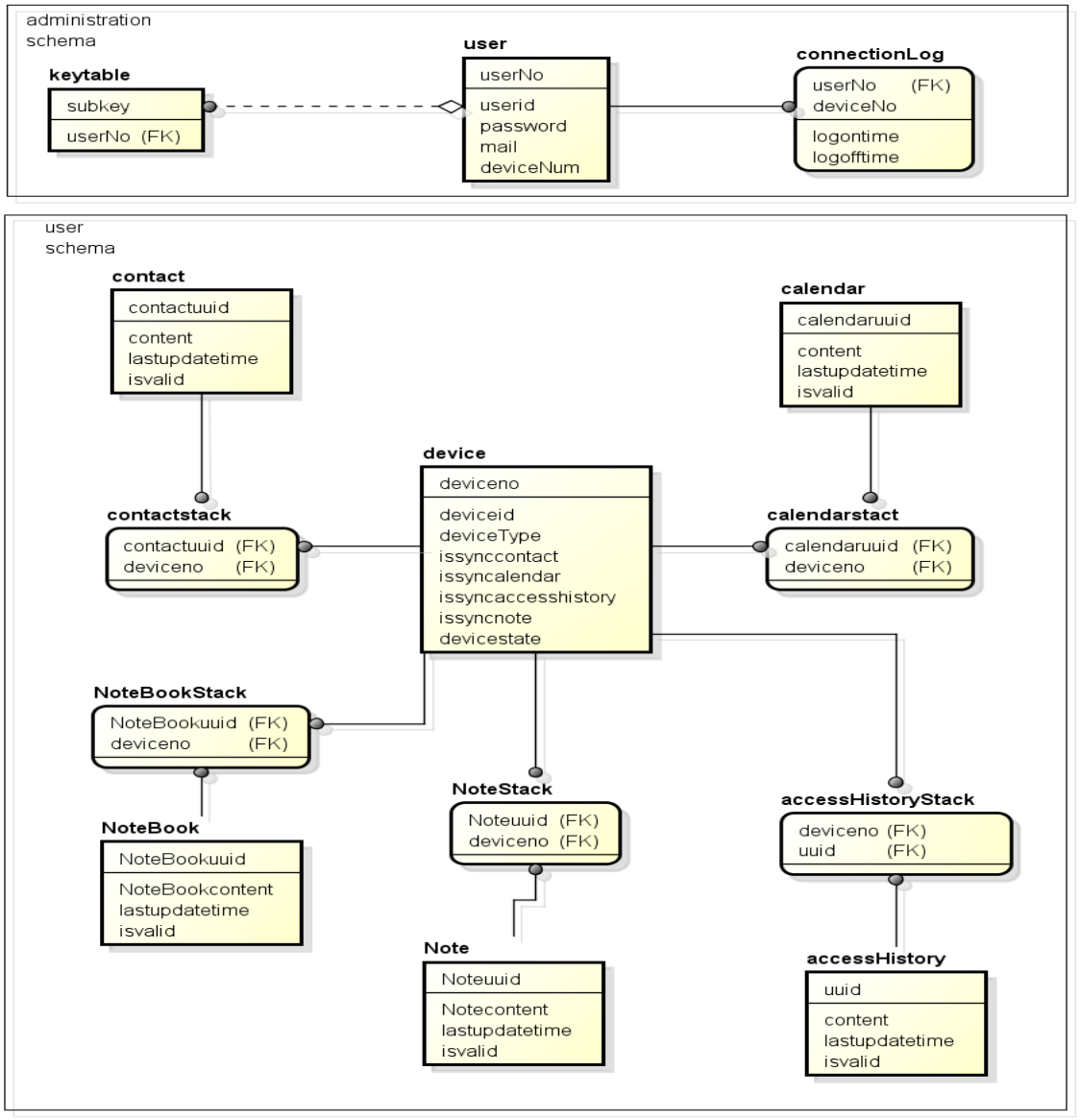
2.3　エンティティー覧

エンティティ ID	エンティティ名	説明
EN001	ユーザー情報	ユーザーに関する情報を管理する
EN002	デバイス情報	デバイスに関する情報を管理する
EN003	コンタクト情報	コンタクト情報を管理する
EN004	履歴情報	履歴に関する情報を管理する
EN005	カレンダー情報	カレンダーに関する情報を管理する
EN006	ノートブック情報	ノートブックの情報を管理する
EN007	ノート情報	ノートの情報を管理する
EN008	コンタクト stack	デバイスがオフライン時、同期必要があるコンタクトの情報を管理する
EN009	履歴 stack	デバイスがオフライン時、同期必要がある履歴の情報を管理する
EN010	カレンダーstack	デバイスがオフライン時、同期必要があるカレンダーの情報を管理する
EN011	ノートブック stack	デバイスがオフライン時、同期必要があるノートブックの情報を管理する
EN012	ノート stack	デバイスがオフライン時、同期必要があるノートの情報を管理する
EN013	サブキー情報	データベースを暗号化するためのキーを保存する場所
EN014	ログ	システムを利用する行為の記録を管理する

2.4　論理 ER 図



2.5　物理 ER 図



2.6　エンティティ定義

2.6.1　管理者スキーマ情報

・ユーザー情報

エンティティ ID	EN001							
エンティティ名	ユーザー情報							
説明	ユーザーに関する情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	ユーザーナンバー	userNo	PK (auto increment)		整形	10	NG	ユーザーの実体を一意に識別する番号
2	ユーザーID	Userid			バイナリデータ		NG	システムに登録する時、利用される
2	メールアドレス	Mail			文字列	320	OK	ユーザーのメールアドレス
3	パースワード	Password			バイナリデータ		NG	ユーザーのパースワード

・サブキー情報

エンティティ ID	EN013							
エンティティ名	サブキー情報							
説明	ユーザーに関する情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	ユーザーナンバー	userNo		FK	整形	10	NG	ユーザーの実体を一意に識別する番号

2	サブキー	subKey			文字列	50	NG	ユーザーに属するサブキー

・ログ

エンティティ ID	EN014							
エンティティ名	接続ログ							
説明	時間区分に関する情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	ユーザーナンバー	userNo		FK	整数	10	NG	ユーザーの実体を一意に識別する番号
2	デバイスナンバー	deviceNo			整数	10	NG	デバイスの実体を一意に識別する番号
3	ログオン時間	logontime			年月日時分秒		NG	ユーザーのデバイスをログオンの時刻
4	ログアウト時間	logouttime			年月日時分秒		NG	ユーザーのデバイスをログアウトの時刻

2.6.2 ユーザースキーマ情報

・デバイス情報

エンティティ ID	EN002							
エンティティ名	デバイス情報							
説明	デバイスに関する情報 を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	デバイスナンバー	no	PK (auto increment)		整数	10	NG	デバイスの実体を一意に識別する番号
2	デバイス ID	deviceid			バイナリ		NG	デバイスのマックアドレス

					データ			
3	種類	deviceType			文字列	20	ok	デバイスの種類
4	カレンダー同期可否	issynccontact			論理		NG	カレンダー同期するかどうかを判断するための値
5	コンタクト同期可否	Issynccalendar			論理		NG	コンタクト同期するかどうかを判断するための値
6	履歴同期可否	issynchistory			論理		NG	履歴同期するかどうかを判断するための値
7	ノート同期可否	Issyncnote			論理		NG	ノートとノートブックを同期するかどうかを判断するための値
8	接続状態	Devicestate			論理		NG	デバイスの接続状態

・コンタクト情報

エンティティ ID	EN003							
エンティティ名	コンタクト情報							
説明	コンタクト情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	コンタクト ID	uuid	PK		文字列	50	NG	コンタクトを一意に識別する番号
2	内容	content			バイナリデータ		NG	コンタクトに関するすべての具体的なデータを単一の個体として保存する
3	lastUpdateTime	lastupdatetime			年月日時分秒		NG	最後更新した時刻
4	有効	Isvalid			論理		NG	データの有効性

・履歴情報

エンティティ ID	EN004
エンティティ名	履歴情報

説明	履歴情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	履歴 ID	uuid	PK		文字列	50	NG	履歴を一意に識別する番号
2	内容	content			バイナリデータ	500	NG	履歴に関するすべての具体的なデータを単一の個体として保存する
3	lastUpdateTime	lastupdatetime			年月日時分秒		NG	最後更新した時刻
4	有効	Isvalid			論理		NG	データの有効性

・カレンダー情報

エンティティ ID	EN005							
エンティティ名	カレンダー情報							
説明	カレンダー情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	カレンダーID	uuid	PK		文字列	50	NG	カレンダーを一意に識別する番号
2	内容	content			バイナリデータ		NG	カレンダーに関するすべての具体的なデータを単一の個体として保存する
3	lastUpdateTime	lastupdatetime			年月日時分秒		NG	最後更新した時刻
4	有効	Isvalid			論理		NG	データの有効性

・ノートブック情報

エンティティ ID	EN006							
エンティティ名	ノートブック情報							
説明	ノートブック情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明

1	ノートブック ID	uuid	PK		文字列	50	NG	ノートブックを一意に識別する番号
2	内容	content			バイナリデータ		NG	ノートブックに関するすべての具体的なデータを単一の個体として保存する
3	lastUpdateTime	lastupdatetime			年月日時分秒		NG	最後更新した時刻
4	有効	Isvalid			論理		NG	データの有効性

・ノート情報

エンティティ ID	EN007							
エンティティ名	ノート情報							
説明	ノート情報を管理する							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	ノート ID	uuid	PK		文字列	50	NG	ノートを一意に識別する番号
2	内容	content			バイナリデータ		NG	ノートに関するすべての具体的なデータを単一の個体として保存する
3	lastUpdateTime	lastupdatetime			年月日時分秒		NG	最後更新した時刻
4	有効	Isvalid			論理		NG	データの有効性

・コンタクト stack

エンティティ ID	EN008							
エンティティ名	コンタクト stack							
説明	**デバイスがオフライン時、同期必要があるコンタクトの情報を管理する**							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	デバイスナンバー	deviceNo	PK	FK	整数	10	NG	デバイスの実体を一意に識別する番号

2	コンタクト ID	contactid	PK	FK	文字列	50	NG	コンタクトを一意に識別する番号

・履歴 stack

エンティティ ID	EN009							
エンティティ名	履歴 stack							
説明	**デバイスがオフライン時、同期必要がある履歴の情報を管理する**							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	デバイス ID	deviceNo	PK	FK	整数	10	NG	デバイスの実体を一意に識別する番号
2	履歴 ID	Historyid	PK	FK	文字列	50	NG	履歴を一意に識別する番号

・カレンダーstack

エンティティ ID	EN010							
エンティティ名	カレンダーstack							
説明	**デバイスがオフライン時、同期必要があるカレンダーの情報を管理する**							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	デバイス ID	deviceNo	PK	FK	整数	10	NG	デバイスの実体を一意に識別する番号
2	カレンダー ID	Calendared	PK	FK	文字列	50	NG	カレンダーを一意に識別する番号

・ノートブック stack

エンティティ ID	EN011							
エンティティ名	ノートブック stack							
説明	**デバイスがオフライン時、同期必要がある**ノートブック**の情報を管理する**							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明

1	デバイス ID	deviceNo	PK	FK	整数	10	NG	デバイスの実体を一意に識別する番号
2	ノートブック ID	Notebookid	PK	FK	文字列	50	NG	ノートブックを一意に識別する番号

・ノート stack

エンティティ ID	EN012							
エンティティ名	ノート stack							
説明	**デバイスがオフライン時、同期必要があるノートの情報を管理する**							
項番	属性	物理名	主キー	外部キー	データ型	桁数	ヌル値	説明
1	デバイス ID	deviceNo	PK	FK	整数	10	NG	デバイスの実体を一意に識別する番号
2	ノート ID	noteid	PK	FK	文字列	50	NG	ノートを一意に識別する番号

2.6.3 特記事項

1. ユーザーごとにスキーマを作成する。
2. コンタクト stack、履歴 stack、カレンダーstack、ノートブック stack とノート stack の内容は同期完成する時点で、テーブルの項目を削除する

コーディング基準書（クライント側）

作成者：
改訂者：
作成日：
改訂日：

目　次

1. はじめに

1.1. 概要

本資料は、クライント側のコーディング形式に関する指針を示したものである。クライント側のソフトウェア開発担当者は、開発作業を開始する前準備として本資料の内容を熟知しておく必要がある。

コーディング基準を設定する主な理由は、アプリケーションの構造とコーディングスタイルを標準化することにより、コードを簡単に理解できるようにすることである。適切なコーディング基準を設定することで、正確で読みやすく、あいまいな部分のないソース コードになる。また、他の言語規則との一貫性を持たせることができ、直観的にもわかりやすくなる。

2. 全体

2.1. 行文字数

1 行に記述可能な文字数に関しては制限を設けないものとする。

2.2. コメント記述言語

コメントの記述には基本的に日本語を使用する。

2.3. その他

1. クラスは最初の文字は大文字で使う。
2. グローバルメソッドや変数を使わない。
3. ローカル変数が定義されたら、使用前にかならず初期化とする。
4. 一つのクラスをひとつのファイルにする。
5. ファイル名とクラス名は同じである。
6. メソッドや属性の中の名前は「_」を使わない。

3. ファイルヘッダー

3.1. ファイルヘッダー

1)フォーマット
ファイルヘッダーの記述方法を下記の通り定める。
② バージョンを記述する。
②機能概要や目的を記述する。
2) インクルード
同じファイルを重複インクルードするを防ぐため、#ifndef の定義は必要である。例えば contact.h ファイルは以下の定義が必要である。

```
#ifndef  CONTACT_H
#define CONTACT_H
 … …
 #endif //CONTACT_H
```

順番はまずライブラリのファイルを先にインクルードして、次に自分で作成したファイルをインクルードする。

4. コメント

プログラムを理解しやすいため、ある程度コメントを書く必要ある。その一方、コメント量は多すぎるやなさすぎる場合は、邪魔になるわけである。よって、コメントはプログラム全体の 20%～30%ぐらいに目安とする。しかも、コメントはできるだけ短く正しく書くようにとする。

4.1. コメント方法

1．　クラス、インタフェース、パブリックメソッドを必ずコメントし、機能や目的を説明する。
　ファイルヘッダには改訂履歴を必ず記述する。
2．　複雑なロジックで実現されるメソッドには、実現方法を説明する。
3．　ソースの右や上にコメントを記述する。
4．　ソースとコメントと一致する。ソースを更新したら、必ずコメントも一緒に更新する。
5．　コメントにはソースコードと関係ないと内容を記述しない。

5. ネーミング

5.1. クラス

1．　最初の文字を大文字にする。
2．　複数の単語をくみあわせれば、全て単語の最初の文字を大文字にする。
3．　名詞と形容詞で記述する。動詞は使わない。
4．　クラスの責務をできるだけ反映させる。例えば class Config,class Contact。
5．　親クラス名と子クラス名の関連性を示す。

5.2. メソッド

1．　最初の文字を小文字にする。
2．　できるだけ動詞と名詞を組み合わせて記述する。例えば bool addContact(),bool removeContact()。
3．　メッソドの操作や目的を反映する。

5.3. 属性

1．大文字と小文字の両方を使用し、各単語の先頭を大文字にする。
2．［形容詞］+ 名詞 の形式で記述する。

6. 原則

6.1. 正確性原則

プログラムはまず正しく実行できる原則が必要である。少なくとも以下のことを守るべき。

1．　クラスの生成と削除において、new と delete を使う。
2．　変数を使う前に、かならず初期化する。
3．　一行で一つの変数を定義する。
4．　複雑な制御文やポイント文を使わない。例えば goto や**p。
5．　Switch 文に必ず default 文がある。
6．　エラーを防ぐため、定数と変数の同じかどうかを比較する時に、定数は前に書く。

例えば if(0 == value)

```
{
… …
}
```

6.2. 保守性原則

プログラムはある程度保守性を確保する必要である。

1．　ロジック同じのスコープはできうるだけ重複しない。その内容を一つのメソッドにまとめる。
2．　メソッドをなさすぎないようにする。できるだけメソッドは 100 行を超えない。
3．　メソッド名と実現内容は一致する。
4．　メソッドの責務はシンプルである。一つのメソッドを複数のタスクを実装しない。
　例えば RemoveAndAddContact()は以下のように分けて実装する。
　RemoveContact();
　AddContact();
5．　オーペンソースをできるだけ修正しない。

6.3. 移植性原則

1．　特定のプラットフォームに関わる部分をかからない部分を分けて実装する。
2．　できるだけ、char,short,int,long などの基本データ型を使わない。
3．　特定のデイバスに依存部分と非依存部分を分けて書く。

6.4. 効率性原則

1．　多重 if 制御文のある場合は一番忙しくない if 制御文を再優先に実装すべき。
2．　タスクの多い場合は多重スレッドで並列実行するようにする。
3．　プログラムの実行スピードよりプログラムのアーキテクチャを優先に実施する。

6.5. テスト可能原則

1．　全てプログラムがテスト可能である。
2．　Debug 文やブレックポイントを使って、プログラムの正しさをチェックする。
3．　リリース版は全て Debug 文を実行しないようにする。

7. その他

1．　プログラムを作成したら、必ず自分でチェックする。
2．　プログラムを各前に必要最小限設計を作成した。

コーディング基準書（サーバー側）

作成者：
改訂者：
作成日：
改訂日：

目　次

1. はじめに

1.1. 概要

本資料は、サーバー側のコーディング形式に関する指針を示したものである。サーバー側のソフトウェア開発担当者は、開発作業を開始する前準備として本資料の内容を熟知しておく必要がある。

コーディング基準を設定する主な理由は、アプリケーションの構造とコーディングスタイルを標準化することにより、コードを簡単に理解できるようにすることである。適切なコーディング基準を設定することで、正確で読みやすく、あいまいな部分のないソース コードになる。また、他の言語規則との一貫性を持たせることができ、直観的にもわかりやすくなる。

2. 全体

2.1. 行文字数

1 行に記述可能な文字数に関しては制限を設けないものとする。

2.2. コメント記述言語

コメントの記述には簡潔に記述すること。

メソッド：

```
/// <summary>
/// 処理名。。
/// </summary>
/// <param name="aaa"></param>
/// <param name="bbb"></param>
/// <returns></returns>
```

ソースの中：

```
//
```

また、/*　*/は使わないとする

2.3. その他

1. クラスは最初の文字は大文字で使う。
2. グローバルメソッドや変数を使わない。
3. ローカル変数が定義されたら、使用前にかならず初期化とする。
4. 一つのクラスをひとつのファイルにする。
5. ファイル名とクラス名は同じである。

3. コメント

プログラムを理解しやすいため、ある程度コメントを書く必要ある。その一方、コメント量は多すぎるやなさすぎる場合は、邪魔になるわけである。よって、コメントはプログラム全体の 20%～30%ぐらいに目安とする。しかも、コメントはできるだけ短く正しく書くようにとする。
また、ライブラリの場合は、プロジェクトの出力で、コメントの XML ファイルを設定する。

3.1. コメント方法

1. クラス、インタフェース、パブリックメソッドを必ずコメントし、機能や目的を説明する。
2. 複雑なロジックで実現されるメソッドには、実現方法を説明する。
3. ソースの上にコメントを記述する。
4. ソースとコメントと一致する。ソースを更新したら、必ずコメントも一緒に更新する。
5. コメントにはソースコードと関係ないと内容を記述しない。

4. ネーミング

4.1. クラス

1. 最初の文字を大文字にする。
2. 複数の単語をくみあわせれば、全て単語の最初の文字を大文字にする。
3. 名詞と形容詞で記述する。動詞は使わない。
4. クラスの責務をできるだけ反映させる。例えば class Config,class Contact。
5. 親クラス名と子クラス名の関連性を示す。

4.2. メソッド

1. 最初の文字を小文字にする。
2. できるだけ動詞と名詞を組み合わせて記述する。例えば bool addContact(),bool removeContact()。
3. メッソドの操作や目的を反映する。

4.3. 属性

1. 大文字と小文字の両方を使用し、各単語の先頭を大文字にする。
2. ［形容詞］+ 名詞 の形式で記述する。

5. 原則

5.1. 正確性原則

プログラムはまず正しく実行できる原則が必要である。少なくとも以下のことを守るべき。

1. 変数を使う前に、かならず初期化する。
2. 一行で一つの変数を定義する。
3. 複雑な制御文やポイント文を使わない。例えば goto や**p。

5.2. 保守性原則

プログラムはある程度保守性を確保する必要である。

1. ロジック同じのスコープはできうるだけ重複しない。その内容を一つのメソッドにまとめる。
2. メソッド名と実現内容は一致する。
3. メソッドの責務はシンプルである。一つのメソッドを複数のタスクを実装しない。
   例えば RemoveAndAddContact()は以下のように分けて実装する。
   ```
   RemoveContact();
   AddContact();
   ```
4. オーペンソースをできるだけ修正しない。

5.3. 効率性原則

1．　多重 if 制御文のある場合は一番忙しくない if 制御文を再優先に実装すべき。
2．　タスクの多い場合は多重スレッドで並列実行するようにする。
3．　プログラムの実行スピードよりプログラムのアーキテクチャを優先に実施する。

5.4. テスト可能原則

1．　全てプログラムがテスト可能である。
2．　Debug 文やブレックポイントを使って、プログラムの正しさをチェックする。

6. その他

1．　プログラムを作成したら、必ず自分でチェックする。
2．　プログラムを各前に必要最小限設計を作成した。

資料名	**単体テスト項目表**
サブシステム名	**サーバープログラム**
クラス名	**_ConnectionManager**
ファイルのバージョン	**V1.0**

項目	メソード名	入力	期待結果	テスト結果	概要
1	_Connection Manager	_	内部用関数と表示用データテーブルは初期かされた	OK	初期化を確認
2	Start	通常	サーバーは起動された	OK	サーバー起動の確認
3		DB異常	サーバーはエラーを返した	OK	
4	CreateManagerSocket	port利用できる	通常成功した	OK	
5		他のプログラムはportを使用中	エラーを返し	OK	
6	doManageConnection	通常、新接続きた場合	セッション作成された	OK	デバイスから新たの接続に対して、処理されたかを確認
7		通常、接続切った場合	表示用テーブルから削除	OK	画面表示用データテーブルの確認
8		異常の場合	ログに記録された	OK	ログは記録されたかを確認
9	doCreateNewSession	新たの接続が来た場合	表示用テーブルに追加された	OK	画面表示用データテーブルの確認
10	doSendDataToSession	他のデバイスへ送信命令があった場合	他のデバイスにデータを送信した	OK	セッション間通信を確認
11	GetSessionIdFromDeviceId	他のデバイスへ送信する前	デバイスIDからセッションIDを取得された	OK	デバイスIDとセッションIDの関連付けを確認
12	doProcessXMLStream	データ受信した時	XML処理は呼ばれた	OK	
13	stop	ユーザは画面でボタンを押下した	サーバーは終了した	OK	サーバー終了の確認
14	CloseSocket	特定のソケットを閉じる	ソケットは閉じた	OK	不要のソケットは解放したかを確認
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	_Session
ファイルのバージョン	V1.0

項目	メソード名	入力	期待結果	テスト結果	概要
1	_Session		クラス初期化完了	OK	初期化の確認
2	Close		セッション終了 利用したリソースを解放した	OK	接続切れた時にて確認
3	doProcessXMLStream		XMLは処理されたこと	OK	XML処理インスタンスに処理データを渡す
4	doBeginReceive		受信準備完了	OK	受信準備完了したか、受信できるかを確認
5	GetPassedSeconds		経過時間を返す	OK	経過した時間は正しく取得できるかどうかを確認
6	SendDataCallBack		CallBackは呼び出されること	OK	送信完了した時呼び出されたかどうかを確認
7	ReceiveDataCallBack		CallBackは呼び出されること	OK	受信完了した時呼び出されたかどうかを確認
8	doProcessBuffer	ソケットデータ	新データはバッファに追加されたこと	OK	一時的データはバッファに保存されたかを確認
9	OnSendDataToOtherSession	送信データ	サーバーインスタンスにデータを送たこと	OK	他のデバイスに送信する時に確認
10	SendToDevice	送信データ	デバイスにデータを送信したこと	OK	デバイスに正しく送信したかを確認
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	_SecurityTransportManager
ファイルのバージョン	V1.0

項目	メソード名	入力	期待結果	テスト結果	概要
1	InitRSA		RSAインスタンスは初期化された	OK	RSA初期化を行って、publickey privatekeyは作成されたかを確認
2	InitAES		AESインスタントを初期化された	OK	
3	AESEncrypt		バイト配列に対して暗号化された	OK	暗号化することを確認
4	AESDecrypt		バイト配列データは復号された	OK	復号化することを確認
5	GetRandom Str		動的に文字列を生成された	OK	
6	GetMd5		文字列のMD5結果を取得できた	OK	MD5処理を確認
7	ConvertHex StringToByt eArray		HEX文字列のバイト配列を取得できた	OK	クライアントから受信したHEX文字列のバイトは配列データの転換を確認
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	_XMLOperator
ファイルのバージョン	V1.0

項目	メソード名	入力	期待結果	テスト結果	概要
1	doProcessXMLStream	ログイン指示 正確データ	ログインチェックを行う、OKの結果を返す	OK	ログインチェックの正確性を確認
2	doProcessXMLStream	ログイン指示 不正データ	ログインチェックを行う、NGの結果を返す	OK	ログインチェックの正確性を確認
3	doProcessXMLStream	パスワード変更指示 正確データ	パスワード変更を行う、OKの結果を返す	OK	パスワード変更処理の正確性を確認
4	doProcessXMLStream	接続指示 正確データ	接続を行う、OKの結果を返す	OK	接続処理の正確性を確認
5	doProcessXMLStream	ユーザ作成指示 正確データ	ユーザ作成を行う、OKの結果を返す	OK	ユーザ作成処理の正確性を確認
6	doProcessXMLStream	パスワード変更指示 不正データ	パスワード変更を行う、NGの結果を返す	OK	パスワード変更処理の正確性を確認
7	doProcessXMLStream	接続指示 不正データ	接続を行う、NGの結果を返す	OK	接続処理の正確性を確認
8	doProcessXMLStream	ユーザ作成指示 不正データ	ユーザ作成を行う、NGの結果を返す	OK	ユーザ作成処理の正確性を確認
9	doProcessXMLStream	データ更新指示 正確データ	データ更新を行う、OKの結果を返す	OK	データ更新処理の正確性を確認
10	doProcessXMLStream	データ更新指示 不正データ	データ更新を行う、NGの結果を返す	OK	データ更新処理の正確性を確認
11	GetData	XMLストリーム	XMLデータを取得し	OK	XML外リームからデータ取得できるかを確認
12	InitContactTable		コンタクトデータ保存用データテーブルは初期かされた	OK	データテーブル初期化の正確性を確認
13	InitzNoteTable		ノートデータ保存用データテーブルは初期かされた	OK	データテーブル初期化の正確性を確認
14	InitNoteBookTable		ノートボックデータ保存用データテーブルは初期かされた	OK	データテーブル初期化の正確性を確認
15	InitCalendarTable		カレンダーデータ保存用データテーブルは初期かされた	OK	データテーブル初期化の正確性を確認
16	InitHistoryTable		履歴データ保存用データテーブルは初期かされた	OK	データテーブル初期化の正確性を確認
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	_ConfigReader
ファイルのバージョン	V1.0

項目	メソード名	入力	期待結果	テスト結果	概要
1	GetValue	ファイル存在しない	エラーコード返す	OK	iniファイルは存在しない時エラーコードは返すかどうかを確認
2	GetValue	ファイル存在する	IP、Port無事に取得できる	OK	通常の場合、IP、Portは取得できることを確認
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	**単体テスト項目表**
サブシステム名	**サーバープログラム**
クラス名	**_Logger**
ファイルのバージョン	**V1.0**

項目	メソード名	入力	期待結果	テスト結果	概要
1	Write	ログ記録しる場合	ログファイルに記録された	OK	ログファイルに、ログ内容は記録されるかを確認
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	DB操作モジュール
クラス名	SystemDBOpeater
ファイルのバージョン	V1.0

項目	メソード名	入力	期待結果	概要	
1	Initiate	データベースに接続する文字列	ok	データベースに接続	
2	AddUser	testuser,1234,raa@gmail.com,on11	ok	データベースに新しいユーザーを追加すること	
3	AddUser	testuser,zsczx,abc@gmail.com,on2	ok	既存なuserNameを入力すると、エラーコードを出力する	
4	DeleteUser	testuser1	ok	正しいユーザ名を利用し、ユーザー	
5	DeleteUser	aaaa	問題発生	存在しないユーザー名を利用する場	
6	ChangePassword	testuser,2222	ok	正しいユーザー情報を利用し、ユーザーのpasswordを変更する	
7	ChangePassword	aaaaa ,2313	問題発生	存在しないユーザー名を使用すると、エラーコードを出力する	
8	GetUserDB	testuser	ok	正しいユーザー名を利用し、ユー	
9	GetUserDBOperator	aaaaa	ok	存在しないユーザー名を使用すると、エラーコードを出力する	
10	UserLogon	testuser,2222,on1111,isNewDevic	ok	正しいユーザー情報を利用し、かつ既存のデバイスを利用し、正しく登録	
11	UserLogon	testuser,2222,on2222,isNewDevice	ok	正しいユーザー情報を利用し、かつ新しいデバイスを利用し、正しく登録でき、デバイスをデータベースに追加	
12	UserLogon	testuser,23123,on1111,isNewDevi	ok	間違えたpasswordを使用すると、エラーコードを出力する	
13	UserLogon	aaaa,1234,on1111,isNewDevice	問題発生	存在しないユーザー名を使用すると、エラーコードを出力する	
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					

資料名	単体テスト項目表
サブシステム名	DB操作モジュール
クラス名	UserDBOpeater
ファイルのバージョン	V1.0

項目	メソード名	入力	テスト結果	概要	
1	Open	なし	ok	正しいのDBConnectionを生成する	
2	GetDeviceIds	deviceIds	ok	ユーザーに属するデバイスを取得する	
3	GetOnlineD	onLineDeviceIds	ok	ユーザーに属するオンラインデバイ	
4	GetOfflineD	offLineDeviceIds	ok	ユーザーに属するオフラインを取得	
5	GetSyncProperty	testdevice,contacts,calendar,accessHistory	ok	正しいデバイスIDを利用し、同期可否を取得する	
6	GetSyncPro	aaaa,contacts,cal	ok	存在しないデバイスを使用すると、エ	
7	UpdateDeviceState	testdevice,deviceState	ok	正しいデバイスを利用し、デバイスの接続状態をオンラインにする	
8	UpdateDeviceState	aaaa,deviceState	ok	存在しないデバイスを使用すると、エラーコードを出力する	
9	SetSyncProperty	testdevice contacts,calendar,accessHistory	ok	正しいデバイスを利用し、デバイスの同期可否を設定する	
10	SetSyncProperty	aaaa,contacts,calendar,accessHistory	ok	存在しないデバイスを使用すると、エラーコードを出力する	
11	GetContactLastUpdateTime	testcontact,lastUpdateTime	ok	正しいcontactのidを利用し、コンタクトの最後の更新時間を取得する	
12	GetContactLastUpdateTime	aaa,lastUpdateTime	ok	存在しないコンタクトのidを利用すると、エラーコードを出力する	
13	GetAllContacts	contacts	ok	ユーザーが所有するすべてのコンタクトを取得する	
14	AddNewContact	testcontact,contactContents,lastUpdateTime	ok	データベースに新規のコンタクトを追加する	
15	UpdateContact	testcontact,contactContents,lastupdateTime	問題発生	正しいコンタクトのidを利用し、コンタクトの内容を更新する	
16	UpdateContact	aaa,conactContents,lastUpdateTi	ok	存在しないコンタクトのIdを利用すると、エラーコードを出力する	
17	DeleteContact	testcontact	ok	正しいcontactのidを利用し、コンタクトを無効にする	
18	DeleteContact	aaa	問題発生	存在しないデバイスを使用すると、エラーコードを出力する	
19	AddContactToStack	testdevice,testcontact	ok	有効なコンタクトIDを利用し、contactstackにデータを追加する	
20	AddContactToStack	aaa,testcontact	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
21	GetContactStackData	testdevice,contacts	ok	デバイスに属するすべてのコンタクトを取得する	
22	GetContact	aaa,contacts	ok	存在しないデバイスIDを利用すると、	
23	DeleteContactStack	testdevice	ok	デバイスに属するすべてのコンタクトを削除する	
24	DeleteCont	aaa	ok	存在しないデバイスIDを利用すると、	
25	GetNoteBookLastUpdateTime	testnoteBook,lastUpdateTime	ok	正しいNoteBookのidを利用し、ノートブックの最後の更新時間を取得する	
26	GetNoteBookLastUpdateTime	aaa,lastUpdateTime	ok	存在しないノートブックのidを利用すると、エラーコードを出力する	
27	GetAllNoteBooks	noteBooks	ok	ユーザーが所有するすべてのノートブックを取得する	

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	UserDBOpeater
ファイルのバージョン	V1.0

項目	メソード名	入力	テスト結果	概要	
28	AddNewNoteBook	testnoteBook,noteBookContents,l	ok	データベースに新規のノートブックを追加する	
29	UpdateNoteBook	testnoteBook,noteBookContents,l	ok	正しいノートブックのidを利用し、ノートブックの内容を更新する	
30	UpdateNoteBook	aaa,conactContents,lastUpdateTime	ok	存在しないノートブックのIdを利用すると、エラーコードを出力する	
31	DeleteNoteBook	testnoteBook	ok	正しいノートブックのidを利用し、ノートブックを無効にする	
32	DeleteNoteBook	aaa	ok	存在しないデバイスを使用すると、エラーコードを出力する	
33	AddNoteBookToStack	testdevice,testnoteBook	ok	有効なノートブックIDを利用し、notebookstackにデータを追加する	
34	AddNoteBookToStack	aaa,testnoteBook	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
35	GetNoteBookStackDat	testdevice,noteBooks	ok	デバイスに属するすべてのノートブックを取得する	
36	GetNoteBookStackDat	aaa,noteBooks	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
37	DeleteNoteBookStack	testdevice	ok	デバイスに属するすべてのノートブックを削除する	
38	DeleteNote	aaa	ok	存在しないデバイスIDを利用すると、	
39	GetNoteLastUpdateTim	testcontact,lastUpdateTime	ok	正しいNoteのidを利用し、ノートの最後の更新時間を取得する	
40	GetNoteLastUpdateTim	aaa,lastUpdateTime	ok	存在しないノートのidを利用すると、エラーコードを出力する	
41	GetAllNotes	contacts	ok	ユーザーが所有するすべてのノートを取得する	
42	AddNewNote	testnote,noteContents,lastUpdate	ok	データベースに新規のノートを追加する	
43	UpdateNote	testnote,noteContents,lastupdateT	ok	正しいノートブックのidを利用し、ノートの内容を更新する	
44	UpdateNote	aaa,conactContents,lastUpdateTi	ok	存在しないノートのIdを利用すると、エラーコードを出力する	
45	DeleteNote	testnote	ok	正しいノートのidを利用し、ノートブッ	
46	DeleteNote	aaa	ok	存在しないデバイスを使用すると、エ	
47	AddNoteToStack	testdevice,testnote	ok	有効なノートIDを利用し、notestackにデータを追加する	
48	AddNoteToStack	aaa,testnote	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
49	GetNoteStackData	testdevice,notes	ok	デバイスに属するすべてのノートを取得する	
50	GetNoteSta	aaa,notes	ok	存在しないデバイスIDを利用すると、	
51	DeleteNoteStack	testdevice	ok	デバイスに属するすべてのノートを削除する	
52	DeleteNote	aaa	ok	存在しないデバイスIDを利用すると、	
53	GetAllCalendars	Calendars	ok	ユーザーが所有するすべてのカレンダーイベントを取得する	
54	AddNewCalendar	testCalendar,CalendarContents,lastUpdateTime	ok	データベースに新規のカレンダーイベントを追加する	

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	_ConfigReader
ファイルのバージョン	V1.0

項目	メソード名	入力	テスト結果	概要	
55	UpdateCalendar	testCalendar,CalendarContents,la	ok	正しいカレンダーイベントブックのidを利用し、カレンダーイベントの内容を	
56	UpdateCale	aaaCalendartCon	ok	存在しないカレンダーイベントのIdを	
57	DeleteCalendar	testCalendar	ok	正しいカレンダーイベントのidを利用し、カレンダーイベントブックを無効に	
58	DeleteCalendar	aaa	ok	存在しないデバイスを使用すると、エラーコードを出力する	
59	AddCalendarToStack	testdevice,testCalendar	ok	有効なカレンダーイベントIDを利用し、Calendarstackにデータを追加す	
60	AddCalendarToStack	aaa,testCalendar	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
61	GetCalendarStackData	testdevice,Calendars	ok	デバイスに属するすべてのカレンダーイベントを取得する	
62	GetCalenda	aaa,Calendars	ok	存在しないデバイスIDを利用すると、	
63	DeleteCalendarStack	testdevice	ok	デバイスに属するすべてのカレンダーイベントを削除する	
64	DeleteCalen	aaa	ok	存在しないデバイスIDを利用すると、	
65	GetHistoryLastUpdateTime	testHistory,lastUpdateTime	ok	正しいHistoryのidを利用し、閲覧履歴の最後の更新時間を取得する	
66	GetHistoryLastUpdateTime	aaa,lastUpdateTime	ok	存在しない閲覧履歴のidを利用すると、エラーコードを出力する	
67	GetAllHistorys	Historys	ok	ユーザーが所有するすべての閲覧履歴を取得する	
68	AddNewHistory	testHistory,HistoryContents,lastUpdateTime	ok	データベースに新規の閲覧履歴を追加する	
69	UpdateHistory	testHistory,HistoryContents,lastupdateTime	ok	正しい閲覧履歴のidを利用し、閲覧履歴の内容を更新する	
70	UpdateHistory	aaa,conactContents,lastUpdateTi	ok	存在しない閲覧履歴のIdを利用すると、エラーコードを出力する	
71	DeleteHistory	testHistory	ok	正しい閲覧履歴のidを利用し、閲覧履歴を無効にする	
72	DeleteHistory	aaa	ok	存在しないデバイスを使用すると、エラーコードを出力する	
73	AddHistoryToStack	testdevice,testHistory	ok	有効な閲覧履歴IDを利用し、Historystackにデータを追加する	
74	AddHistoryToStack	aaa,testHistory	ok	存在しないデバイスIDを利用すると、エラーコードを出力する	
75	GetHistoryStackData	testdevice,Historys	ok	デバイスに属するすべての閲覧履歴を取得する	
76	GetHistoryS	aaa,Historys	ok	存在しないデバイスIDを利用すると、	
77	DeleteHistoryStack	testdevice	ok	デバイスに属するすべての閲覧履歴を削除する	
78	DeleteHisto	aaa	ok	存在しないデバイスIDを利用すると、	

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	Encryption
ファイルのバージョン	V1.0

項目	メソード名	入力	テスト結果	概要	
1	generateMai	testuser	ok	ユーザーIDのMD5値を生成する	
2	DecryptSub Key	testuser	ok	既存なユーザーIDを利用し、ユーザーに属するサブキーを復号化する	
3	DecryptSub Key	aaaaaa	ok	存在しなユーザーIDを利用すると、エラーコードを出力する	
4	generateKe yForUser	testuser	ok	ユーザーIDを利用し、ユーザーに属するサブキーを生成する	

資料名	単体テスト項目表
サブシステム名	サーバープログラム
クラス名	CreateUserSchema
ファイルのバージョン	V1.0

項目	メソード名	入力	テスト結果	概要	
1	inite	データベースに	ok	データベースに接続	
2	createUser SchemaByU	testuser,1234,raa @gmail.com,on11	ok	データベースに新しいユーザーを追加すること	
3	createUser SchemaByU	testuser,zsczx,ab c@gmail.com,on2	ok	既存なuserNameを入力すると、エラーコードを出力する	
4	getUserNo	testuser1	ok	正しいユーザ名を利用し、ユーザーナンバーを獲得する	
5	getUserNo	aaaa	ok	存在しないユーザー名を利用する場	

資料名	単体テスト項目表
サブシステム名	クライント通信処理
クラス名	network
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	結果	概要
1	connectToMeegoServer	サーバーアドレスとパート		サーバーとつながるかどうかを確認
2	disconnectFromServer			通信が切断するか確認
3	displayError			通信エラーが発生した時に、エラーメッセージが現れる
4	sockedConnected			sockekをつながったと時に、受信バッファークリア
5	socketReadyRead			受信完了した時に受信処理を行う
6	sendContactChect			サーバーに送信すべくコンタクトがあるかどうかチェック
7	socketConnected			サーバーと接続したどうか状態を確認
8	socketDisconnected			disconnected状態を確認
9	handleStream			サーバーと接続した後に、clientConnecttionRequest
10	sendDeviceIdInfo			サーバーと暗号化処理を終わったら、デバイスIDをサー
11	addUser			設定されたユーザー名とパスワードをサーバーに送信
12	changeInfo			設定されたユーザー名とパスワード、新しいパスワード
13	logIn			設定されたユーザー名とパスワードをサーバーに送信
14	sendMessage			keepAliveメッセージをサーバーに送信したどうか確認
15	ScanContact			コンタクトのデータを取得したかどうか確認
16	sendContact	contactXML		送信すべくコンタクトのデータをサーバー送信したか確
17	contacttoXML			コンタクトのデータを標準XMLに変換したかどうか確認
18	receiveDataResult			サーバーに送られたデータを正しいか確認
19	sendData	送信すべく文字列		送信すべくデータをすべでサーバーに送信したか確認
20	createTempStr	10		暗号化要の文字列を生成したかどう確認
21				
22				
23				
24				

テスト実施日										
テスト実施時間										
実施者										
実施結果										
障害処理票ID										

資料名	単体テスト項目表
サブシステム名	クライント内部処理
クラス名	XMLpacket
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	結果	概要
1	commandParse	beginConnectionCommand XMLメッセージ		XMLメッセージを解析して、コマンドを取得したか確認
2	commandParse	userLogonCommand XMLメッセージ		XMLメッセージを解析して、コマンドを取得したか確認
3	commandParse	serverSendDataCommandXMLメッセージ		XMLメッセージを解析して、コマンドを取得したか確認
4	commandParse	addUserCommandXMLメッセージ		XMLメッセージを解析して、コマンドを取得したか確認
5	commandParse	userInfoUpdateCommandXMLメッセージ		XMLメッセージを解析して、コマンドを取得したか確認
6	resultParse	beginConnectionCommand XMLメッセージ		XMLメッセージを解析して、codeを取得したか確認
7	resultParse	userLogonCommand XMLメッセージ		XMLメッセージを解析して、codeを取得したか確認
8	resultParse	addUserCommandXMLメッセージ		XMLメッセージを解析して、codeを取得したか確認
9	resultParse	userInfoUpdateCommandXMLメッセージ		XMLメッセージを解析して、codeを取得したか確認
10	contactParse	＜contact＞<uuid>{12345678-1234-1234-123456781234}</uuid><uuid></uuid><content>abcd</content><lastupdatetime></lastupdatetime><IsValid>0</IsValid>＜/contact＞		XMLメッセージを解析してコンタクトノデータを取得したか確認
11	contactParse	＜contact＞<uuid>{12345678-1234-1234-123456781234}</uuid><uuid></uuid><content>abcd</content><lastupdatetime></lastupdatetime><IsValid>1</IsValid>＜/contact＞		XMLメッセージを解析してコンタクトノデータを取得したか確認
12	contactToXml	＜contact＞<uuid>{abcdefig-1234-1234-123456781234}</uuid><uuid></uuid><content>abcd</content><lastupdatetime></lastupdatetime><IsValid>1</IsValid>＜/contact＞		コンタクトの文字列をXMLに正しく変換したか
13	contactToXml	＜contact＞<uuid>{abcdefig-1234-1234-123456781234}</uuid><uuid></uuid><content>abcd</content><lastupdatetime></lastupdatetime><IsValid>0</IsValid>＜/contact＞		コンタクトの文字列をXMLに正しく変換したか
14				
15				
16				
17				
18				
19				
20				

21	
22	
23	

テスト実施日										
テスト実施時間										
実施者										
実施結果										
障害処理票ID										

資料	単体テスト項目表
サブシステム名	クライント内部処理
クラス名	timethread
ファイルのバージョン	V1.0（実機）

項目	メソード名	入力・操作	結果	概要
1	copyContactstoDBCopy	最新のコンタクトデータ		最新のコンタクトデータをDBCCopyFileにコーピーする確認
2	editContactsCopy	DBCopyデータとサーバから受信されたデータ		サーバーからコンタクトによって、ContactCopyを更新、同じUUIdであれば書き換え、新しいIdであればContactCopyに追加を確認
3	readContactsFromLocal	コンタクトにあ新しいデータを追加		MeeGo　Contactから追加データを読み取る確認
4	readContactsFromLocal	コンタクトにデータを削除		MeeGo　Contactにデータが削除され、読み取ったデータを確認
5	readContactsFromLocal	コンタクトにデータを変更		MeeGo　Contactにデータが削除され、読み取ったデータを確認
6	addContacts	コンタクトに追加すべきデータ		追加すべきデータをコンタクトに追加する確認
7	rewriteContacts	コンタクトに編集すべきデータ		編集すべきデータをコンタクトに上書きする確認
8	removeContacts	コンタクトに削除すべきデータ		削除すべきデータをコンタクトに削除する確認
9	comparecontacts	コンタクトのデータとDBCopyのデータ		コンタクトに新しいデータを追加してから、比較する確認
10	comparecontacts	コンタクトのデータとDBCopyのデータ		コンタクトに既存のデータを修正してから比較する確認
11	comparecontacts	コンタクトのデータとDBCopyのデータ		コンタクトに既存のデータを削除してから比較する確認
12	comparecontactAfterReceive	コンタクトデータと送信されたデータ		サーバから同じUuidのデータをもらってkら比較する確認
13	comparecontactAfterReceive	コンタクトデータと送信されたデータ		サーバから無効のデータをもらってから比較する確認
14	comparecontactAfterReceive	コンタクトデータと送信されたデータ		サーバから新しいデータをもらってから比較する確認
15	compareTime	localTime,copyTime		2つのデータの時間比較を確認
16	ContactsFileAddtoFile	追加すべくコンタクト		追加すべくデータをファイルに追加したどうか確認
17	removeContactsFromFile	削除すべくコンタクト		削除すべくデータをファイルに削除したどうか確認
18	rewriteContactsToFile	変更すべくコンタクト		変更すべくデータをファイルに変更したどうか確認
19	readContactsFromFile			ファイルからすべてデータを取得するかどうか確認
20				
21				
22				
23				
24				
15				
16				
17				
18				

テスト実施日									
テスト実施時間									
実施者									
実施結果									
障害処理票ID									

資料名	単体テスト項目表
サブシステム名	クライント内部処理
クラス名	config
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	結果	概要
1	setHost	192.168.11.3		入力したIPアドレスを設定したか確認
2	getHost			設定したIPアドレスを取得したか確認
3	setPort	3130		入力したポート番号を設定したか確認
4	getPort			設定したポート番号を取得したか確認
5	setpassword	1234		入力したパスワードを設定したか確認
6	getpassword			設定したパスワードを取得したか確認
7	setDeviceId			入力したデバイスIDを設定したか確認
8	getDeviceId			設定したデバイスIDを取得したか確認
9	setNewPassword	123456		入力した新しいパスワードを設定したか確認
10	getNewPassword			設定した新しいパスワードを取得したか確認
11	setConnectionType	1		入力した接続タイプを設定したか確認
12	getConnectionType			設定した接続タイプを取得したか確認
13	setConnectionType	2		入力した接続タイプを設定したか確認
14	setConnectionType	3		入力した接続タイプを設定したか確認
15				
16				
17				
18				
19				

テスト実施日										
テスト実施時間										
実施者										
実施結果										
障害処理票ID										

資料名	単体テスト項目表
サブシステム名	クライント内部処理
クラス名	DBCopySave
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	結果	概要
1	DBCopyDirCreate			/meego/home/DBCopyのディレクトリを作成したか確認
2	ContactCopyFileCrea			/meego/home/DBCopy/ContactCopyFile.txtを作成したか
3	contactFileWrite	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23 10:5:23.0		入力されたコンタクトデータをContactCopyFile.txtに書き込んだか確認
4	ContactFileRead			ContactCopyFile.txtの内容を読み取ったか確認
5	ContactFileClearAll			
6	ContactFileRemove	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		入力したデータを削除したか確認
7	ContactFileRewrite	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		同じUuidのデータをファイルに上書きたか確認
8	ContactFileAdd	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		入力したデータをファイルに追加したか確認
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

テスト実施日										
テスト実施時間										
実施者										
実施結果										
障害処理票ID										

資料名	単体テスト項目表
サブシステム名	クライント内部処理
クラス名	Contact
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	結果	概要
1	readContacts			コンタクトのすべてデータを取得したか確認
2	addContacts	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		入力したデータをコンタクトに追加したか確認
3	removeContacts	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		入力したデータをコンタクトに削除したか確認
4	rewriteContacts	{12345678-1234-1234-123456781234},abcd,abcd,abcd,,,,2011-2-23		同じUuidのデータをコンタクトに上書きたか確認
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

テスト実施日										
テスト実施時間										
実施者										
実施結果										
障害処理票ID										

資料名	単体テスト項目表
サブシステム名	Note共有機能
クラス名	Note
ファイルのバージョン	V1.1（実機対応）

項目	メソード名	入力	テスト結果	概要
1	timerFunc	NoteBookデータ	更新されたデータの	NoteBookデータのXML生成確認
2	timerFunc	XMLデータが合計9999文字の	分割のないXMLデータが生成	分割が行われない場合のNoteデータのXMLデータの生成確認
3	timerFunc	本文のデータが10000文字の	2つに分割された更新用XMLデータを生	分割が1回行われる場合のNoteデータのXML生成確認(境界値)
4	timerFunc	本文が20000文字場合のNoteデータの更新	3つに分割された更新用XMLデータの生成	分割が2回以上行われる場合のNoteデータのXML生成確認
5	timerFunc	本文を20000文字から19999文字へ更新	2つに分割された更新用データと1つの削除用XMLデータの	分割数が減った場合に通常のXMLと分割減少分の削除XMLが生成されるかの確認
6	timerFunc	本文20000文字から9999文字へ更新	分割のない更新用XMLと2つの削除用XMLデータの生成	分割数が減った場合に通常のXMLと過去に分割されたXMLの削除が行われるかの確認
7	timerFunc	更新なし	XML生成なし	更新が行われなかった場合の処理の確認
8	timerFunc	NoteBookデータの削除	削除用XML生成	NoteBookデータを削除した場合のXML生成確認
9	timerFunc	前回分割されていないnoteデータの削除	削除用XML生成	分割がない場合のnoteデータの削除用XML生成の確認
10	timerFunc	前回分割されたnoteデータの削除	分割数分の削除用XML生成	分割がある場合のnoteデータの削除
11	updateData	更新日時が1秒新しいNoteBookのXMLデータ	noteBooksテーブルの更新	NoteBookデータを受信したときの更新処理の確認
12	updateData	更新日時が等しいNoteBookのXMLデータ	更新なし	更新の必要がないデータを受信した場合の処理の確認
13	updateData	分割されておらず更新日時が1	notesテーブルの更新	分割がない場合のnoteデータの更新処理の確認
14	updateData	分割されておらず更新日時が等しいNoteのXML	更新なし	更新の必要がないデータを受信した場合の処理の確認
15	updateData	1回分割されており、かつ更新日時が同一で分割分がそろっているNoteのXMLデー	結合された状態でのnotesテーブル更新splitItems,splitDataテーブルの更新なし	分割がある場合の結合処理とデータ更新の確認
16	updateData	1回分割されており、かつ更新日時が同一で分割分がそろっていないNoteのXMLデータ	notesテーブルの更新なしsplitItems,splitDataテーブルの更新	分割数がそろっていない場合に更新されないこと、及び更新されなかったデータが保存されることの確認
17	updateData	分割数に満たないデータがsplitItemsテーブルに入っている状態で更新日時のより新しい分割データの受信	splitItemsの既存データの削除、新規データの追加、splitDataテーブルの更新	分割数がそろっていない状態で新しいデータを受信した場合の処理の確認
18	updateData	分割数に満たないデータがsplitItemsテーブルに入っている状態で更新日時が古い分割デー	更新なし	更新すべきでない分割データを受信した場合の処理の確認

19	updateData	分割数に満たないデータがsplitItemsテーブルに入っている状態でデータの削除XML	splitItems、splitDataテーブルから該当データの削除	分割途中に削除された場合の確認
20	updateData	noteBooksテーブルに所属する	noteBookIdが0の状態でnotesテーブル	所属するNoteBookが存在しない場合のNoteの受信
21	updateData	Noteが存在するNoteBookデータの受信	NoteのnoteBookId更新 テーブルから該当データの削除	所属するNoteBookが後から送信されてきた場合の処理
22	updateData	Noteが存在する場合の所属するNoteBookの変更XMLの受信	notesテーブルの該当データのnoteBookId更新	所属するNoteBookが存在しないnotesの所属するNoteBookが変更された場合の処理
23	updateData	Noteが存在する場合の削除用XML受信	notesテーブルの該当データ削除 テーブルの該当データ削除	所属するNoteBookが存在しないnotesが削除された場合の処理
24	init	データベースが存在しない	データベース全体の作成	Noteアプリケーションが一度も起動されていない場合の処理
25	init	データベースは存在するがデータは入っておら	Logデータベースとトリガの作成	Noteアプリケーションは実行されたことがあるがNoteデータが作られたことがなく、共有システムが実行されたことのない場合の処理
26	init	データベースは存在し、かつデータは入っているが	Logデータベースとトリガの作成 存在するデータの	Noteアプリケーションが実行されており、かつNoteデータが既に存在する場合の処理
27	init	データベースは存在し、Logデータベースも存在する	処理なし	既に一度実行済みの場合の処理

資料名	結合テスト項目表
サブシステム名	サーバープログラムとデータベース
クラス名	–
ファイルのバージョン	V1.0

項目	入力	期待結果	テスト結果	概要
1	ユーザ登録指示	DBにが追加されること	OK	テストクライアントプログラムも用いてテストする ユーザ登録できることを確認
2		処理結果をクライアントへ返すこと	OK	
3	ユーザパスワード変更指示	変更されたパスワードはDBに変更されること	OK	テストクライアントプログラムも用いてテストする ユーザパスワード変更できることを確認
4		処理結果をクライアントへ返すこと	OK	
5	新しいデバイスIDで接続指示	DBに新しいデバイスIDを追加されること	OK	テストクライアントプログラムも用いてテストする 新しいデバイスIDで接続できることを確認
6		接続結果をクライアントに返すこと	OK	
7	既存デバイスIDで接続指示	デバイス状態はオンラインになること	OK	テストクライアントプログラムも用いてテストする 既存デバイスIDで接続できることを確認
8		接続結果をクライアントに返すこと	OK	
9	ユーザログイン	ログイン結果と更新すべきデータはクライアントに返すこと	OK	テストクライアントプログラムも用いてテストする ユーザログインできることを確認
10	接続切断	デバイス状態はオフラインになること	OK	テストクライアントプログラムも用いてテストする 接続切断した時、正しく処理できることを確認
11	同期データ送信	データをDBに保存されること	OK	テストクライアントプログラムも用いてテストする 同期データ処理できることを確認
12		処理結果をクライアントへ返すこと	OK	
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				

資料名	結合テスト項目表
サブシステム名	UI・内部処理間
クラス名	Note
ファイルのバージョン	V1.1（実機対応）

項目	メソッド名	入力	テスト結果	概要
1	turnOnSwitch	正しいid,passを入力してログイン	サーバーに接続される	ログイン処理の確認
2	turnOnSwitch	登録されていないid,passを入力してログイン	接続が失敗する	誤った入力をした場合のログイン処理の確認
3	adduser	登録されていないそれぞれ10文字のid,passを入力して	ユーザがサーバに登録され	ユーザ登録処理の確認
4	adduser	登録されているid,passを入力してユーザ登録		存在するidに対してユーザ登録しようとした場合の確認
5	changeInfo	登録されているid,passを入力してパスワード変更		パスワード変更処理の確認
6	changeInfo	登録されていないid,passを入力してパスワード変更		誤った入力をした場合のログイン処理の確認
7	turnOnSwitch	id,pass未入力の状態でログイン		入力がない状態でログイン処理をしようとした場合の処理の確認
8	adduser	id,pass未入力の状態でユーザ登録		入力がない状態でユーザ登録しようとした場合の処理の確認
9	changeInfo	id,pass未入力の状態でパスワード変更		入力がない状態でパスワード変更しようとした場合の処理の確認
10	turnOnSwitch	idのみ入力した状態でログイン		パスワードの入力がない状態でログイン処理をしようとした場合の処理の確認
11	adduser	idのみ入力した状態でユーザ登録		パスワードの入力がない状態でユーザ登録処理をしようとした場合の処理の確認
12	changeInfo	idのみ入力した状態でパスワード変更		パスワードの入力がない状態でパスワード変更をしようとした場合の処理の確認
13	turnOnSwitch	passのみを入力した状態でログイン		idの入力がない状態でログインしようとした場合の処理
14	adduser	passのみを入力した状態でユーザ登録		idの入力がない状態でユーザ登録しようとした場合の処理
15	changeInfo	passのみ入力した状態でユーザ登録		idの入力がない状態でパスワード変更しようとした場合の処理
16	changeInfo	id,passが入力され、新規パスワードが入力されていない状態でパスワード変更		新しいパスワードが設定されていない状態でパスワード変更しようとした場合の処理
17	adduser	idが11文字のデータでユーザ登録		限界値より大きい長さのidを登録しようとした場合の処理
18	adduser	passが11文字のデータをユーザ登録		限界値より大きい長さのpassを登録しようとした場合の処理

資料名	結合テスト項目表
サブシステム名	クライアントソフトウェアとサーバーソフトウェア間
クラス名	
ファイルのバージョン	V1.1（実機対応）

項目	項目名	入力	結果	概要
1	サーバーとの接続		OK	ユーザー登録またはログインする時に、自動的にサーバーと接続できるかどうかを確認
2	20秒毎にKeepAliveメッセージをサーバーに送信		OK	20秒ごとにKeepAliveメッセージを送ったログインの時にユーザIDとパスワードをサーバーに送信するかどうかを確認
3	ユーザーログオン時にサーバーに送信		OK	ユーザーグインする時にユーザIDとパスワードをサーバーに送信するかどうか確認
4	ユーザー登録時に送信		OK	ユーザー新規登録する時にユーザIDとパスワードをサーバーに送信するかどうか確認
5	ユーザーパスワード変更時に送信		OK	ユーザーパスワード変更する時にユーザIDとパスワードをサーバーに送信するかどうか確認
6	コンタクト共有（追加）		OK	タブレットAに新しいコンタクトを追加して、タブレットBに追加したどうかを確認
7	コンタクト共有（変更）		OK	タブレットAにあるコンタクトの内容を変更して、タブレットBにそのコンタクトの内容がタブレットAと同じように変更するかどうかを確認
8	コンタクト共有（削除）		OK	タブレットAにあるコンタクトを削除して、タブレットBにそのコンタクトがタブレットAと同じように削除うされるかどうかを確認
9	ノート共有（追加）		OK	タブレットAに新しいノートを追加して、タブレットBに追加したどうかを確認
10	ノート共有（変更）		OK	タブレットAにあるノートの内容を変更して、タブレットBにそのノートの内容がタブレットAと同じように変更するかどうかを確認
11	ノート共有（削除）		OK	タブレットAにあるノートを削除して、タブレットBにそのノートがタブレットAと同じように削除うされるかどうかを確認
12	ノートブック共有（追加）		OK	タブレットAに新しいノートブックを追加して、タブレットBに追加したどうかを確認
13	ノートブック共有（変更）		OK	タブレットAにあるノートブックのタイトルを変更して、タブレットBにそのノートブックのタイトルがタブレットAと同じように変更するかどうかを確認
14	ノートブック共有（削除）		OK	タブレットAにあるノートブックを削除して、タブレットBにそのコンタクトがタブレットAと同じように削除うされるかどうかを確認
15	履歴共有（追加）			タブレットAで新しいWebサイトに訪問して、タブレットBに履歴が追加したどうかを確認
16	履歴共有（変更）			タブレットAで再度Webサイトに訪問し、タブレットBの同一データの更新日時がタブレットAと同じように変更するかどうかを確認
17	履歴共有（削除）			タブレットAにある履歴を削除して、タブレットBでも同様に削除されるかの確認
18	カレンダー共有（追加）		OK	タブレットAに新しい予定を追加して、タブレットBに追加したどうかを確認

19	カレンダー共有（変更）			タブレットAにある予定の内容を変更して、タブレットBにその予定の内容がタブレットAと同じように変更するかどうかを確認
20	カレンダー共有（削除）		OK	タブレットAにある予定を削除して、タブレットBにその予定がタブレットAと同じように削除うされるかどうかを確認
21				
22				
23				
24				
25				
26				
27				

ID	内容	質問者	解答者	状態
1	return 値は0なのに、実はエラーはが発生したところがあり 一旦、ソース再確認お願いします。	劉宏超	劉建宇	完成
2	仕様追加依頼 int GetUsers(List<string> userNames) 仕様¥DBアクセスライブラリ.xlsx　をご参考ください	劉宏超	劉建宇	完成
3	DeleteContact 不要な場合、flagを1に設定すべき 一般追加した場合やflag未指定で追加する場合、初期0だから（例：AddNewContact） ※現状データ取得するところもすべて変更すべき	劉宏超	劉建宇	完成 →×修正してない →初期設定はデータベー
4	AddDeviceId nullで追加されてる列はいくつからしい、 でも取得時は、trueかfalseで指定するため 初期追加はfalseで、いいのでは ※他のところ、同様なミスはあるかどうかも確認	劉宏超	劉建宇	完成
5	エラー sheet2を参考	劉宏超	劉建宇	完成
6	仕様変更： UserLogonにdeviceIdを追加 新deviceの場合、deviceIdを登録するため 仕様¥DBアクセスライブラリ.xlsx　をご参考ください	劉宏超	劉建宇	完成
7	UpdateDeviceState where条件はないらしい	劉宏超	劉建宇	完成
8	deviceId初期はnull 初期を0（オフラインに変更）	劉宏超	劉宏超	完成
9	仕様追加依頼 GetAllContacts(out DataTable contacts) 仕様¥DBアクセスライブラリ.xlsx　をご参考ください 新ディバイス対応 ユーザコンタクトを全部取得	劉宏超	劉宏超	完成
10	仕様変更依頼 UserLogonに引数追加　　out bool isNewDevice 仕様¥DBアクセスライブラリ.xlsx　をご参考ください 新ディバイスであれば、stackデータでなはく、コンタクト全体返すため	劉宏超	劉宏超	完成
11	UserLogon 存在しないユーザでログオンしようとする場合、 エラーが発生する sheet3参考	劉宏超	劉宏超	完成
12	仕様変更依頼 SystemDBOperatorの Initiate（サーバプログラム起動時呼び出される） に すべてユーザDBのすべてディバイスの状態 (devicestate)を0(オフライン)にする処理を追加 サーバー落ちた（あまり少ないケースが）場合、	劉宏超	劉宏超	完成
13	エラー sheet4を参考	劉宏超	劉宏超	完成
14	GetContactLastUpdateTime 存在しないコンタクトIDで渡す場合、おかしくなる 改造： /// <returns> /// -1:コンタクトは存在しない /// 0:正常 /// 1～:異常 /// </returns>	劉宏超	劉宏超	完成

No.	評価内容		評価結果
	大分類	細分	
1	画面操作	使いやすいか。	OK
2		誤動作防止してるか。(メッセージ表示など)	OK
3	画面表示	各個項目の意味は分かりやすいか。	OK
4		画面表示は適当であるか。	OK
5	システム管理	機能足りるか。	OK
6	負荷能力	300個のクライアント同時に接続できるか。	OK
7	レスポンス	同時に100件データを処理する場合の 処理時間は数秒以外で完了したか。	OK
8		100個クライアント接続中の場合でも正しく初期できるか。	OK
9		同時送信クライアント数　送信データ数	
10		基本　1　　1	OK
11		基本　1　　10	OK
12		基本　1　　100	OK
13		基本　10　　1	OK
14		基本　10　　10	OK
15		負荷　10　　100	OK
16		基本　100　　1	OK
17		負荷　100　　10	OK
18		負荷　100　　100	OK(処理できる)
19	DB I/O処理	基本　1　　1	OK
20		基本　1　　10	OK
21		基本　1　　100	OK
22		基本　10　　1	OK
23		基本　10　　10	OK
24		負荷　10　　100	OK
25		基本　100　　1	OK
26		負荷　100　　10	OK
27		負荷　100　　100	OK

a	評価内容		評価結果
	大分類	細分	
1	ログイン画面	ユーザー登録	OK
2		ユーザーログイン	OK
2		ユーザーパスワード変更	OK
	サーバーと通信	サーバーとの接続	OK
		20秒毎にKeepAliveメッセージをサーバーに送信	OK
		ユーザーログオン時にサーバーに送信	OK
		ユーザー登録時に送信	OK
		ユーザーパスワード変更時に送信	OK
		コンタクトをサーバーに送信	OK
		サーバーから送られたコンタクトを受信	OK
		コンタクトをサーバーに送信	OK
		サーバーから送られたノートを受信	OK
		ノートをサーバーに送信	OK
		サーバーから送られたカレンダーを受信	OK
		カレンダーをサーバーに送信	OK
		サーバーから送られた履歴を受信	OK
		履歴をサーバーに送信	OK
	Contact同期	新しいデータをコンタクトに追加	OK
		コンタクトから指定さらたデータを削除	OK
		コンタクトに指定されたデータを編集	OK
		30秒ごとにコンタクトのデータを取得	OK
	Note同期	新しいデータを追加	OK
		データの削除	OK
		データの編集	OK
		30秒ごとにノートのデータを取得	OK
	履歴同期	新しいデータを追加	OK
		データの削除	OK
		データの編集	OK
		30秒ごとに履歴のデータを取得	OK
	カレンダー同期	新しいデータを追加	OK
		データの削除	OK
		データの編集	OK
		30秒ごとにカレンダーのデータを取得	OK

		6月				7月				8月				9月				10月				11月				12月				1月			
タスク名		6	13	20	27	4	11	18	25	8	15	22	29	5	12	19	26	10	17	24	31	7	14	21	27	5	12	19	26	9	16	23	30
計画立案	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
要件定義	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
クラス設計	計画																																
	実績																																
森本	計画																																
	実績																																
技術調査	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
モジュール作成	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
モジュールテスト	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
デモ作成	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
基礎機能完成	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
機能拡張及び	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
PT_ServerPG	計画																																
	実績																																
劉宏超	計画																																
	実績																																
PT_DB	計画																																
	実績																																
劉建宇	計画																																
	実績																																
PT_ClientPG	計画																																
	実績																																
許先華	計画																																
	実績																																
PT_ClientOther	計画																																
	実績																																
森本	計画																																
	実績																																
IT_Server	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
IT_Client	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
IT_System	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
サーバー評価	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
クライアント評価	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																
報告書作成と発表	計画																																
	実績																																
劉宏超	計画																																
	実績																																
劉建宇	計画																																
	実績																																
許先華	計画																																
	実績																																
森本	計画																																
	実績																																