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クライアントソフトウェアの設計開発－

許　先華

(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

本報告書は、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における特定課題研究「研究開発プロジェクト」で遂行したプロジェクトの報告書である。

本プロジェクトの委託元である日本インテル株式会社では、自社のプロセッサが、モバイルやクラウド・コンピューティングのような急成長中の市場で広く採用されるために Linux ベースのオープンソース・モバイル OS「MeeGo」に全力で取り込んでいる。Nokia などと共同でこの OS を推進してきたが、現状では、まだ日本市場には MeeGo を搭載したモバイル機器は出ていない。モバイル機器メーカーとともに MeeGo の製品化を実現していくことが、今後の課題といえる。そこで筆者は MeeGo をベースにした組込みデバイス向けのシステムの企画立案、及び開発を実施した。

まず、MeeGo デバイス向けのシステム開発を前提として、様々な企画を立案した。具体的にはチームメンバーがそれぞれの企画案を出し、実現性、独創性や有用性などを分析し、検討した。検討の結果は MeeGo デバイス間のデータ共有アプリケーションという案を取り上げた。案を決めた後に、筆者はデータ同期方法と暗号化送受信について、調査を行った。

次にデータ共有システムをプロトタイプによる開発を行った。筆者は開発システム全体のうち、クライアントソフトソフトの開発を担当した。具体的にはクライアントソフトウェアのコンタクト（Contact）同期、内部処理、通信処理モジュールの設計と開発を担当した。コンタクト同期機能は、具体的にコンタクトからデータの取得、コンタクトにデータの追加と削除、変更処理を提供するものである。内部処理機能は、設定と XML 処理、更新監視、比較処理機能から構成される。通信処理モジュールは、OpenSSL 暗号化によるサーバーと Socket 通信を行う機能である。特に内部処理と通信処理部分は汎用性、再利用性を持つ特長がある。

そして、開発したシステムを検証するために、要件定義書に基づく評価試験を行った。結果として、筆者が担当した部分であるデータ共有システムのクライアントソフトウェア（コンタクト同期、内部処理、通信モジュール）については要件定義に沿って実現できた。

目次

図目次

表目次

第1章　はじめに

本報告書は、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における特定課題研究「研究開発プロジェクト」で遂行したプロジェクトの報告書である。本プロジェクトでは、日本インテル株式会社から依頼された MeeGo をベースにした次世代組込みデバイス向けユーザ体験開発を、筆者を含む同専攻高度 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博士前期課程 2 年の学生４名が遂行した。

1.1　前提知識

本プロジェクトの前提知識として、MeeGo、Qt について説明する。

1.1.1　MeeGo

MeeGo[1]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、Intel 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ものである。現在のところ、ネットブックやエントリーレベルのデスクトップ PC、携帯通信機器、車載情報システム、ネットワーク接続テレビ、メディアフォン等の機器を主なターゲットとしている。MeeGo は複数のデバイスに順次対応していく予定であるが、それに伴いアプリケーションも複数のデバイスで利用可能となり、それはアプリケーション開発者にとって大きなメリットになる。

MeeGo のメリットの 1 つは、Atom および ARM の両プロセッサに対応していることである。もう 1 つは、クロスプラットフォームの開発環境「Qt」である。Qt により、デスクトップや組み込み機器向けに高い移植性を確保できる。

1.1.2　Qt

Qt[2]（キュート）は C++言語で書かれたアプリケーション・ユーザインターフェース（UI）フレームワークである。GUI ツールキットとして広く知られている Qt であるが、コンソールツールやサーバーのような非 GUI プログラムでも広く使用されている。Nokia の一部門 Qt デベロップメントフレームワークス社によって開発されている。Qt は、高度なアプリケーションの開発に適した充分な機能、拡張性と移植性の高い API により、アプリケーション開発での効果を最大限に発揮する。 さらに、マルチプラットフォームでも、ひとつのソースコードから複数のプラットフォームで稼働するアプリケーションの開発ができる C++ GUI ツールキットであるので、 Unix、Linux、Windows、Mac OS X 用のアプリケーションをさらに効率よく構築することができる。

1.2　本報告書の構成

本報告書の構成は以下のようになる。

第２章では、本プロジェクトの背景、目的、体制、プロジェクトの遂行と成果について述べる。

第３章では、本プロジェクトで行った企画立案と筆者の企画案について述べる。

第４章では、本プロジェクトで開発するデータ共有システムの目的、機能概要、構成について述べる。

第５章では、開発するシステムにおける関連技術の調査について述べる。
第６章では、担当した部分であるクライアントソフトウェアの設計、実装及び成果物について述べる。
第７章では、開発したシステムの総合評価や担当した部分の評価項目や評価の結果について述べる。
第８章では、プロジェクト進行の工夫と分析について述べる。
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

2.1　　プロジェクトの背景

近年、スマートフォンやタブレットなどデバイスが普及するに従って、それに対応するアプリケーションのニーズが高まっている。特に ios や Android をベースにした様々なアプリケーションは数多く開発されている。

一方、MeeGo をベースにしたアプリケーション開発は少ない。今回のプロジェクトでは、MeeGo の特徴を踏まえて、MeeGo をベースにしたアプリケーションを開発した。

2.2　　プロジェクトの目的

本研究開発プロジェクトは、委託元であるインテル株式会社のビジョンの一つである「コンピューティング・コンテニュアムを実現し、ユーザがデバイスを意識せずにシームレスに活動を行うためのソフトウェアを開発すること」を目的としており、オープンソフトウエアプラットフォームである MeeGo 上で動作するアプリケーションを開発することを目標とした。

2.3　　プロジェクトの進め方

本プロジェクトは、MeeGo をベースにしたアプリケーションの計画立案から、技術調査、アプリケーションの開発、評価までを行った。本プロジェクトの大まかな流れを図 2-1 に示す。

「企画立案」フェーズでは、MeeGo の特徴を活かせる点を始め、多くの視点からアプリケーションを企画し、企画したアプリケーションの中から一つのアプリケーションを絞込み、案を決める。

「技術調査」では、システム設計開発を行う前に、実現手法や利用可能な技術などの調査を行う。

「イテレーション１」においては、バージョン１開発を行い、システムの基本機能を実現する。

「イテレーション２」では、設計やプログラムの修正や機能追加をし、最終版の開発を完成する。その後は評価を行うという形で進めていく。

「評価」フェーズでは、開発されたシステムを用いて、試験を行い、試験の結果からシステムの機能と性能を評価する。

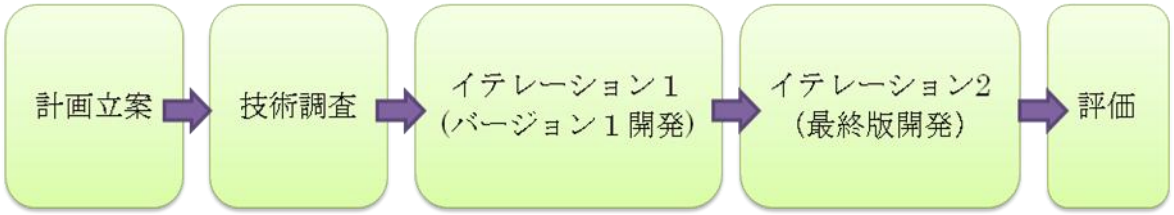


図 2-1　本プロジェクトの進め方

2.4　　プロジェクトの体制

本プロジェクトの開発を図 2-2 に示した体制で行った。学生４名でチームを構成した。筆者は４人チームの一員として、本プロジェクトの企画立案時点から、評価まですべてのフェーズに関わった。

「企画立案」フェーズでは、筆者は家庭用の健康管理システムという案を中心に調査や分析を行った。また他の案について、チームメンバーと一緒に検討した。

「技術調査」では、データ同期技術と通信暗号化技術の調査を行った。

開発段階のイテレーション１フェーズでは、筆者は MeeGo エミュレータ環境をベースにした MeeGo データ同期機能、Socket 通信処理機能、及び内部処理部分機能の一部を実現した。開発段階のイテレーション 2 フェーズでは、筆者は MeeGo タブレット実機をベースにした Contact 同期機能、内部処理機能を完成した。

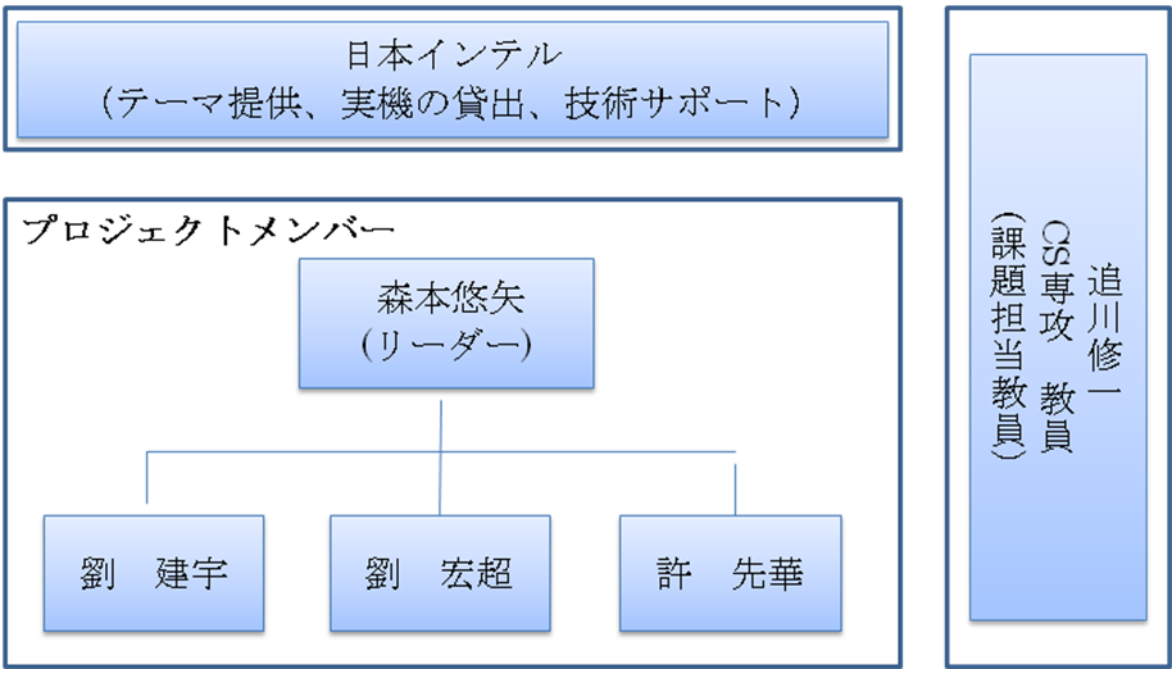


図 2-2　本プロジェクトの開発体制

2.4.1　　プロジェクトの役割分担

本プロジェクトの企画立案フェーズでは、筆者は主に家庭用の健康管理という取組み事案を調査、企画を行った。また、他の案もチームメンバーと一緒に様々な視点から検討した。

開発フェーズでは、本システムのクライアントソフトウェアの設計開発を担当している。具体的にはクライアントソフトウェアのコンタクト同期、内部処理及び通信処理モジュールの開発を担当している。図 2-3 に赤い点線のところに筆者の担当部分である。担当した部分の詳細説明は第 6 章にて詳しく述べる。

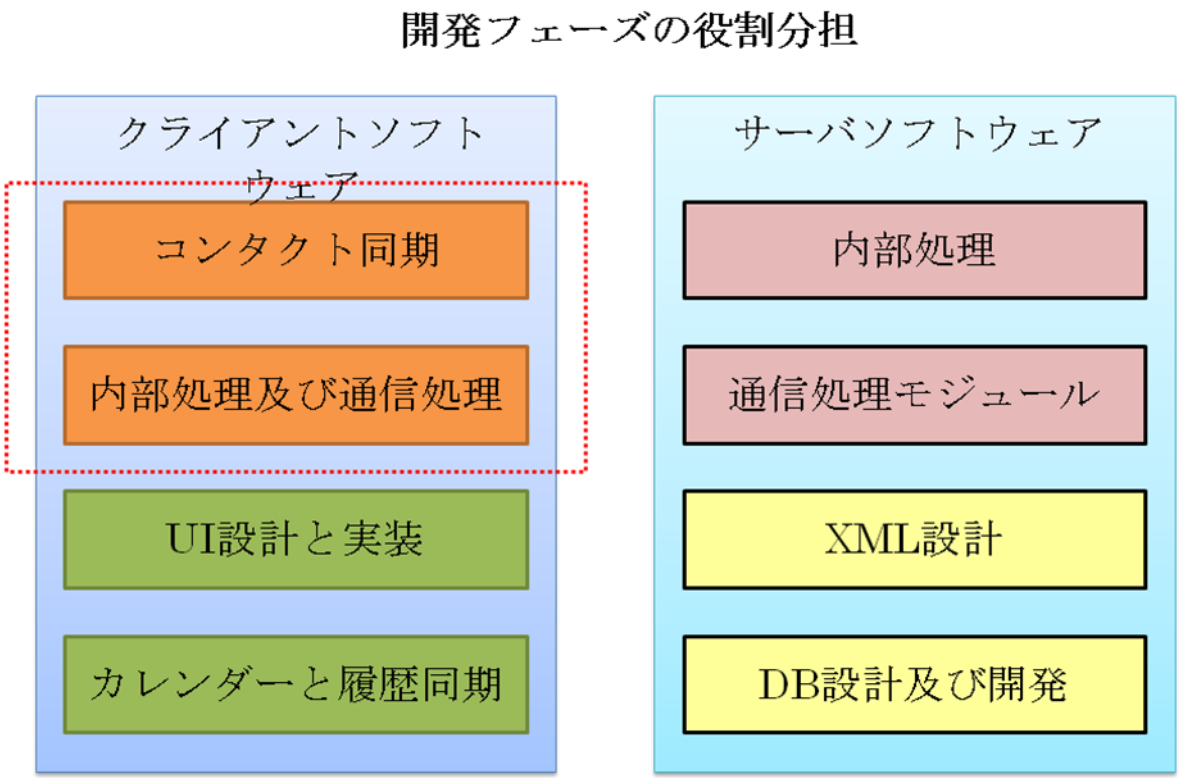


図 2-3　開発フェーズの役割分担図

2.5　　プロジェクトの遂行と成果

本節では、本プロジェクトの計画と実績、成果物を述べる。

2.5.1　　プロジェクトの計画と実績

本プロジェクトはプロトタイプによる開発を進めていく方針である。具体的には MeeGo の基礎技術の学習、案の検討と確定、要件定義、技術調査、モジュール作成と結合、テスト、基本機能完成、評価、機能向上、テスト、評価、報告書作成という順序で行う。プロジェクト全体の工程を表 2-1 に示す。計画と実績を表 2-2 に示す。

表 2-1 プロジェクト全体の工程表

工程名	フェーズ	内容
工程 1	企画立案	基礎技術の習得、環境構築、 アプリケーション案の検討
工程 2	企画立案	アプリケーション確定
工程 3	技術調査 イテレーション 1	技術調査 、 要件定義書作成、クラス設計
工程 4	イテレーション 1	モジュール作成、モジュールテスト、 モジュール結合、結合テスト
工程 5	イテレーション 1	デモ（中間発表）、基礎機能完成
工程 6	イテレーション 2	機能追加、性能向上、テスト
工程 7	イテレーション 2 評価	最終版完成、評価

工程 8		報告書作成

表 2-2 プロジェクトの計画と実績

工程名		6 月	7 月	8 月	9 月	10 月	11 月	12 月	1 月
工程 1	計画								
	実績								
工程 2	計画								
	実績								
工程 3	計画								
	実績								
工程 4	計画								
	実績								
工程 5	計画								
	実績								
工程 6	計画								
	実績								
工程 7	計画								
	実績								
工程 8	計画								
	実績								

・企画立案フェーズは、ほぼ予定通り進んできた。それは MeeGo プラットフォームが多種多様なデバイスに対応できる特徴を活かせる点を始め、さまざまな視点からしっかり企画を立案したためであった。

・技術調査では、予定より長く引いた。特に Buteo、SyncEvolution、QXmpp という同期技術を調べるため、それに関するドキュメントやソースコードを読むのは多くの時間がかかった。

・開発フェーズでは、サーバー側は開発経験が多い Visual C#であるため、計画よりも早く開発が進んだが、クライアント側は Qt と MeeGo プラットフォームに依存しており、Qt と MeeGo の開発経験がなかったため、クライアント側のソフトウェア開発はスケジュールより遅延した。特に事前の MeeGo プラットフォームの技術調査と Qt の勉強が不足したため、モジュールの作成やテストについて、多くの時間がかかり、計画に対して遅延が生じた。

2.5.2 プロジェクトの成果物

プロジェクト全体における成果物とその数量について表 2-3 に示す。

表 2-3 プロジェクトの成果物

工程	成果物	量
要件定義	要件定義書	10 ページ
	ユースケース図	
	ユースケース記述	
基本設計	基本設計書	15 ページ
	概要クラス図及び説明	

	IF 定義書（DB アクセス）	
UI 設計	画面デモ及び定義書（サーバー）	8 ページ
	画面デモ及び定義書（クライアント）	
DB 設計	ER 図	1 ページ
	DB 設計仕様書	17 ページ
詳細設計	詳細設計書（クライアント）	17 ページ
	詳細設計書（サーバー）	16 ページ
	詳細設計書（暗号化）	1 ページ
	詳細設計書（XML 仕様）	8 ページ
	詳細設計書（DB 操作モジュール）	7 ページ
製造	コーディング基準書（クライアント）	5 ページ
	コーディング基準書（サーバー）	5 ページ
テスト	単体テスト仕様書（同期モジュール）	3 ページ
	単体テスト仕様書（DB モジュール）	6 ページ
	単体テスト仕様書（コンタクト、通信と内部処理）	6 ページ
	単体テスト仕様書（サーバー）	6 ページ
	結合テスト仕様書（UI と内部処理）	1 ページ
	結合テスト仕様書（サーバーとクライアント）	1 ページ
	結合テスト仕様書（サーバーと DB）	1 ページ
評価	サーバー評価表	1 ページ
	クライアント評価表	1 ページ

第3章　企画立案

本章では、MeeGo をベースにしたアプリケーションの企画立案について、説明する。

3.1　企画立案の進め方

MeeGo をベースにしたアプリケーション企画立案では、MeeGo の特長を活かせる企画を立案することが目標である。最初にプロジェクトで定めた期間内でできるだけ多くの企画の立案を行う。立案にはブレインストーミングによるアイディアを出しのほか、新規性や機能面、開発面、法律、評価など多方面の視点からの考案ができ、単にアイディアを出すだけに留まらず、実現可能性を視野に入れた企画の立案を行うことができる。

3.2　立案した企画案

最初にチームメンバーはそれぞれに案を出し、独創性や有用性などを分析した上で、以下のように 4 つの案を絞り込んだ。

・家庭用の健康管理
・動画配信
・QML コンポーネントの拡張
・MeeGo Cloud (MeeGo デバイス間のデータ共有システム)

次にその 4 つの提案に対して、機能や課題などを検討した。結果は以下のように示す。

- 家庭用の健康管理

家庭用の健康管理は普段食べている食事や栄養、運動量などを管理するシステムである。利用者の健康管理をサポートする。しかし、問題点としては体重、運動量を計測するためのセンサーは MeeGo デバイス[3]に搭載されていない。そして、医学知識を持っていないと健康生活を維持するために必要な運転量と食物摂取量の分析は難しい。以上の原因で家庭用の健康管理という案を見送った。

- 動画配信

動画配信ではカメラ、マイクなどのデバイスを利用し、生配信や動画会議などのサービスである。その一方、著作権を第三者が有する著作物を保存やアップロードする場合は法的な問題点が起こる可能性がある。

- QML コンポーネットの拡張

QML コンポーネント[4]の拡張は QML コンポーネントを拡張し、新たなボタンなどの UI 部品やアニメーション、文字入力方法などを追加することでソフトウェア開発者がより簡単に MeeGo の UI を構築できるようになることを目指す。ただし、携帯電話向けの MeeGo にはクロスプラットフォームの統合開発環境 Qt IDE はない。そこで MeeGo 携帯をベースにした開発にならない。

- MeeGo Cloud (MeeGo デバイス間のデータ共有システム)

MeeGo Cloud は MeeGo のタブレットや携帯機器、ノートブックを対象にコンタクトやメール、カレンダー、ドキュメント、操作履歴などを共有する。あるデバイスの内容を更新したら、サーバーを経由して他のデバイを同期する。それによって、MeeGo 一つの特徴であるノートブックや携帯機器、車載情報システム、ネットワーク接続テレビなどのデバイスをターゲットとしていることを最大限に発揮することができる。

3.3　　企画の決定

新規性や機能面、開発面、法律、評価など多方面の視点からしっかり検討した上で、MeeGo Cloud（デバイス間のデータ共有アプリケーション）という案を採用した。

3.4　　筆者の企画案

筆者は家庭用の健康管理システムについて、以下のように企画を行った。

◆コンセプト

普段食べている食事や栄養、運動量などを管理するシステムである。利用者は自分の栄養摂取状況、運動量などを把握でき、健康な体を維持するために有用である。

利用者の健康管理をサポートする。具体的には以下のように示す。

（1） 家庭内に設置し、体重、運動量と食物摂取量との関連を記録する。

（2） 記録されたデータに基づいて、健康生活を維持するに必要な運転量と食物摂取量を提示する。

◆処理の流れ

本システムの処理流れを図 3-1 に示す。本システムを利用する際に基本的な情報を登録することが必要である。登録が終わり次第、すぐに利用できる。体重管理、運転量管理、食物摂政量管理などのメニューがある。利用者は自分で好きなメニューを選んで、数値を入力する。システムは入力したデータを基づいて、履歴をグラフで表示する。また、数値の変化を分析し、健康生活を維持するに必要な運転量などを提示する。

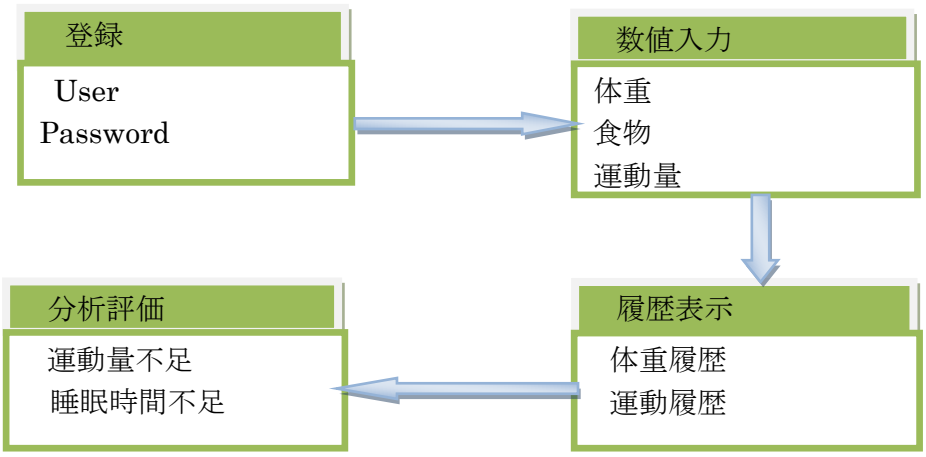


図 3-1　家庭用の健康管理システム処理の流れ

◆問題点と解決方法

入力したデータを基づき、摂取した栄養価や予測される疾病の分析は医学の専門知識が必要となる。その問題を解決するため、利用者は目標値や目標範囲に自分で設定するようにする。設定範囲を超えると提示する。

第4章　　データ共有システムの開発

4.1　　本システムの目的

本システムは MeeGo デバイス上にあるコンタクト、カレンダー、Web の閲覧履歴のよく使われるコンテンツを同一ユーザが持つ複数の MeeGo デバイスに共有することを目的としている。これにより、複数のデバイス間で煩雑な操作をすることなく、自動的に作業を移行することが可能になる。システムの全体構成を図 4-1 に示す。

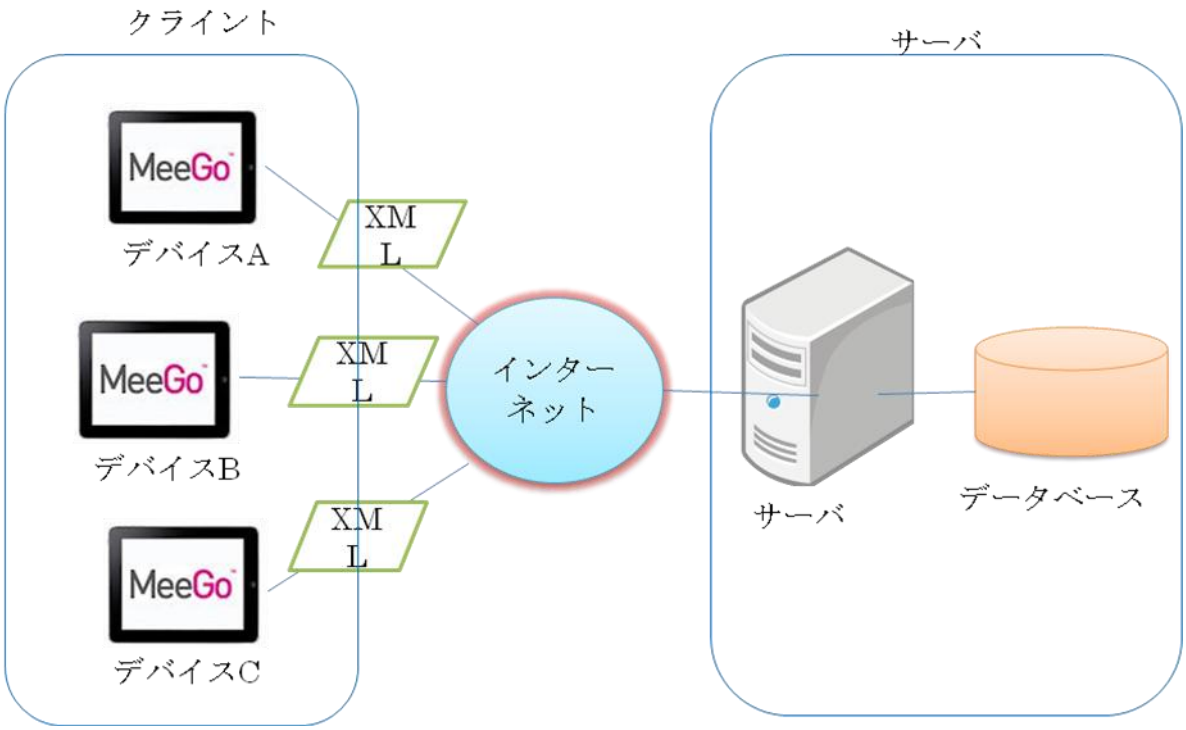


図 4-1　システムの全体構成図

4.2　　本システムの機能概要

本システムは複数の MeeGo デバイス上でコンタクト、カレンダー、Web の閲覧履歴共有する機能を持つ。共有する際に、ユーザ ID とパスワードで認証する必要である。
また、サーバー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 デスクトップに設定パネルを一つ追加し、そのパネル上で操作を行う。全体処理の流れを図 4-2 に示す。

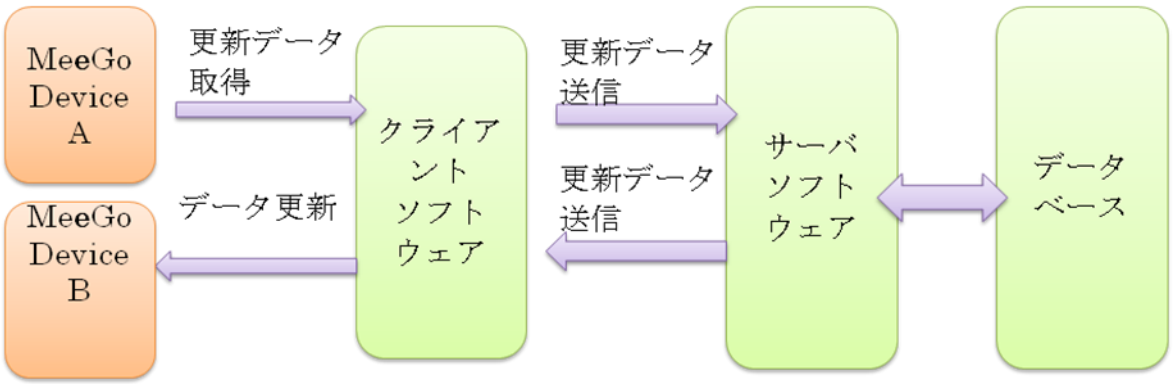


図 4-2　開発したシステム処理の流れ

4.3　本システムの構成

本節は、本システムのハードウェアとソフトウェアの構成について説明する。

4.3.1　ハードウェア構成

ハードウェアの構成要素の説明を次の表 4-1 に示す。

表 4-1　ハードウェアの構成

要素	説明
サーバー	ネットワーク環境を備えたもの。本システムのサーバーとして、同期データの処理とユーザ情報の管理を行う
MeeGo デバイス	ネットワーク環境を備えたもの。本システムのクライアントとして、同期データの処理を行う

現時点では、MeeGo クラウドのクライアントとして利用される MeeGo デバイスは MeeGo タブレットである。

4.3.2　ソフトウェアの構成

MeeGo クラウドの想定動作環境を次の表 4 -2 に示す。

表 4-2　ソフトウェアの構成

種類	名称	バージョン
OS (クライアント)	MeeGo Tablet	1.2.0.90
OS（サーバー）	Windows7 Professional	
DBMS	MySQL	5.15.5

現段階で、バージョン 1.2 以下の OS と MeeGo ハンドセット OS に対応していない。

本システムはクライアント側とサーバー側のソフトウェア開発環境を表 4-3 に示す。サー

バー側のソフトウェアは Windows 上で動作する。データベースは MySQL である。クライアント側のソフトウェアは MeeGo 上で動作する。

表 4-3　本システムの 開発環境

種類	名称	バージョン
クライアント側		
OS	MeeGo Tablet	1.2.0.90
開発言語	C++	
開発環境	Ubuntu Desktop	10.04
	MeeGo SDK	1.2
サーバー側		
OS	Windows7 Professional	
データベース	MySQL	
開発言語	C#	
開発環境	Microsoft Visual Studio 2005	

4.4　本システム要件

4.4.1　機能要件

本節では、実際に開発したシステムの機能要件を紹介する。開発したシステム機能の概要を図 4-3 に示す。ソフトウェアはクライアントとサーバー２つ部分が構成されている。全体機能は以下のように示す。

・ クライアントUI機能
・ クライアントのコンテンツ同期機能（コンタクト、カレンダー、WEBの閲覧履歴）
・ クライアントの内部処理機能
・ クライアントの通信処理機能
・サーバーの通信処理機能
・サーバーの内部処理機能
・サーバーのデータベースアクセス機能
・サーバーのデータベース機能

筆者は担当した機能としては、クライアントの内部処理と通信処理モジュール及びコンタクト同期モジュールである。

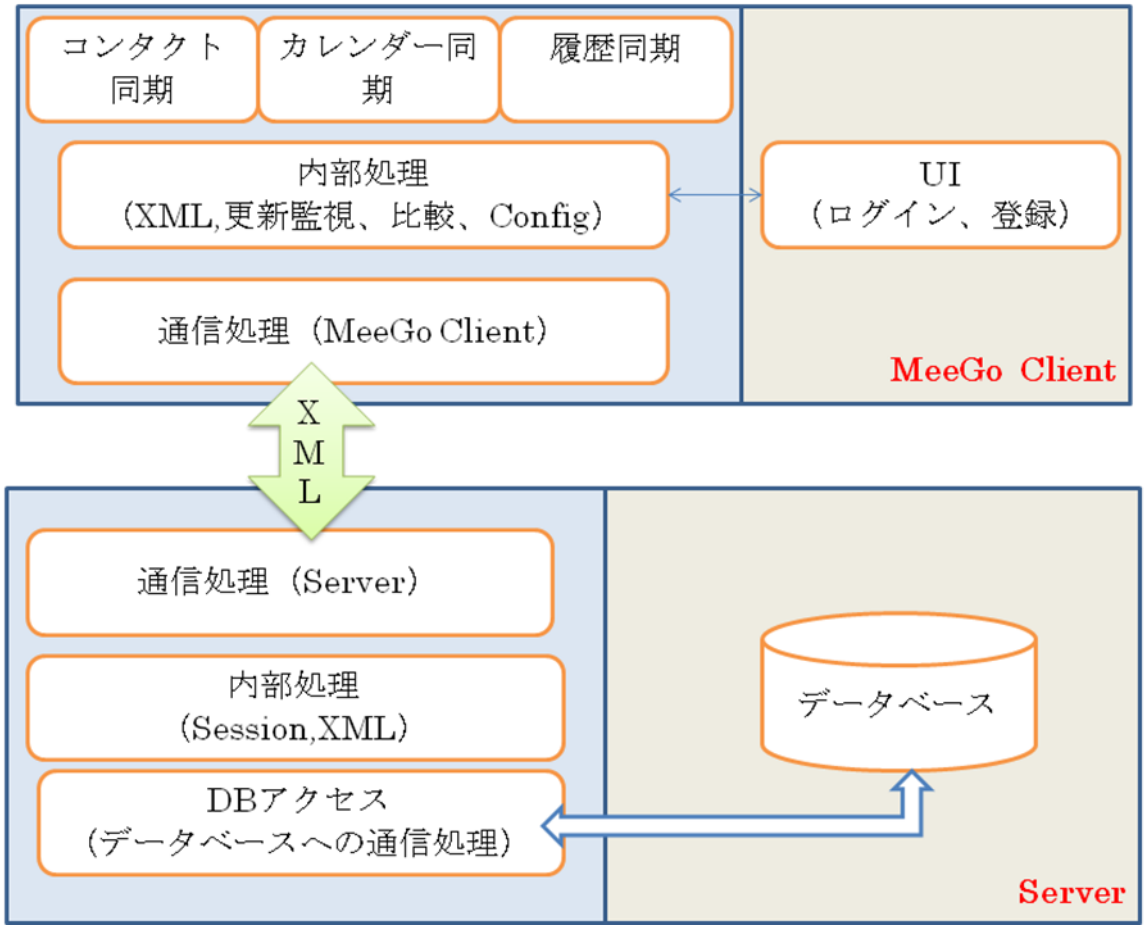


図 4-3　開発したシステム機能の概要図

各モジュールにおいて、具体的に実現すべき機能は以下のようになる。

- クライアントの UI 機能
  UI 部分は ID とパスワードよりサーバーに登録、ログイン及びパスワード変更、共有コンテツの設定を行う為のユーザインターフェースを提供する。
- クライアントのコンテンツ共有機能（コンタクト、カレンダー、閲覧履歴）
  MeeGo のコンテンツをアクセスする機能提供する。具体的にコンテンツからデータの取得、コンタクトにデータの追加や変更という操作を提供する。
- クライアントの内部処理機能
  クライアントの内部処理は具体的に設定、比較処理、XML 処理、更新監視などの部分が構成される。
- クライアントの通信処理機能
  クライアントとサーバー間のやり取りはすべて Soket による通信を行う。また、セキュリディを高めるため、暗号化で通信する。
- サーバーの通信処理機能
  .NET Framework[5] の Socket クラスを使用して、ソケット通信を行う。また、クライアント側と同じ暗号方式で通信する。
- サーバーの内部処理機能
  サーバーの内部処理はあらゆるクライアントの接続管理、クライアントへの送受信やクライアント側から貰った XML 命令の解析・処理などの機能を持つ。

- サーバーのデータベースアクセス機能
  クライアントから新しいデータを貰ったら、データベースに保存する。
- データベース機能
  カレンダー、コンタクトと履歴情報テーブルを設計し、同期コンテンツをそれぞれのテーブルに保存する。また、業務処理によって必要となる DB 処理 API を提供する。

4.4.2 非機能要件

本システムは主に複数デバイス間のデータ共有を行うため、通信の信頼性や効率性は要求される。次に新しいデバイスや機能の追加は容易にする。コンタクト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。

◆信頼性

デバイスが常にネットワークに接続しているとは限らない。また通信途中でネットワークから切断される可能性もある。そこで、途中で切断された場合は、通信が完了したデータのみを更新する。そして複数の更新が見られた場合は最も更新日時が新しいデータとして使う。

◆効率性

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。

◆拡張性と保守性

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。また、通信する際のデータ構造は XML にベースにすることで今後の拡張を容易にする。

◆セキュリティー

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に通信時にはユーザ認証を行う必要がある。また、ユーザ ID とパスワード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。そして、データは全てデータベースに保存し、ユーザごとにデータベース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。

4.5 前提条件と制約事項

本節は、本システムの導入と運営における全体条件と制約条件について述べる。

4.5.1 全体条件

本システムの導入において、以下を前提条件とする。

✧ 対象とするユーザは MeeGo を搭載したタブレットを所有している。

✧ 対象とするユーザは MeeGo タブレットを用いてネットワーク経由して本システムのクライアントソフトウェアをインストールする。

4.5.2 制約事項

本システムの運用において、以下の制約を設ける。

✧ 本システム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。本システムのサーバーは Windows 上で動作する。DB は MySQL の最新版を使用する。

✧ 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ー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ーへのアップロードは行わ

ない。

4.6　本システムの利用シーン

本システムは MeeGo デバイスを持つユーザに対して、デバイス間でのデータ共有機能を利用することができる。図 4-4 にシステム利用際のデータ共有の流れを示す。

① ユーザは MeeGo デバイス A を用いて、サーバーに登録する。
② ユーザは MeeGo デバイス B を用いて、サーバーに登録する。
③ ユーザは MeeGo デバイス A を用いて、コンタクトやカレンダーなどに新しいデータを作成する。
④ システムは自動的にその新しいデータを MeeGo デバイス B に同期する。

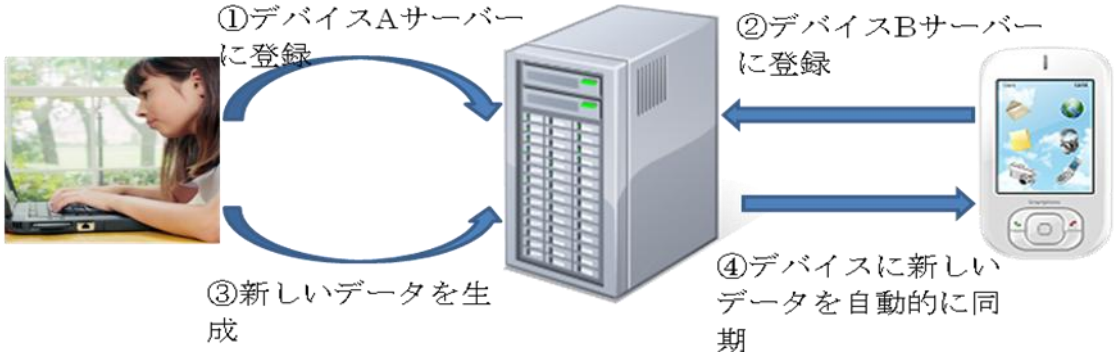


図 4-4　システム利用シーン図

第5章　関連技術の調査

より効率的にクライアントソフトウェアの設計開発を進めるため、設計を行う前にクライアントとサーバーの同期技術や暗号化通信についての詳細を調査する必要がある。よって、筆者はその 2 つ技術を調査した。

5.1　同期技術調査

クライアントソフトウェアのコンタクト同期、内部処理と通信を担当することで、具体的なデータ同期方法や通信プロトコルを参考するため、筆者はその技術を調査した。

・MeeGo Buteo Sync framework[6]

今回利用する MeeGo のバージョンは 1.2 である。そのバージョンの同期フレームワークには Buteo Sync framework が採用されている。しかし、その機能は不十分であり、将来は SyncEvolution[7]で置き換える可能性がある。SyncEvolution とはさまざまなプロトコル（SyncML、CalDAV/CardDAV、ActiveSync）を経由して個人情報管理（PIM）データを同期する既存の成熟したオープンソースフレームワークである。Buteo Sync framework は MeeGo の新しいバージョンで採用されない可能性があるので、今回のプロジェクトでは Buteo Sync framework を利用しないことにした。

・QXMPP[8]

QXMPP は Qt と C++を基づいた同期フレームワークである。開発環境は今回利用される Qt、C++と同じだが、プロトコルは複雑であるため、把握しなければならないプログラムの量は結構多いである。今回の開発期間を考えて、QXMPP を利用するのは難しい。

以上の理由で今回のプログラムは既存の同期フレームワークを利用せずに Qt の API を利用してゼロから新しいクライアント通信ソフトウェアを作る。

5.2　暗号化技術調査

サーバーと通信する際に、セキュリティーを高めるため、暗号化しての送受信は必要である。設計を行う前に筆者は以下の暗号方式を調査した。

・SSL[9] (Secure Sockets Layer)

データを暗号化してやり取りする方法の決まりである。具体的なアルゴリズムはそれぞれ複数の選択肢が定義されており、SSL 通信の開始時に行われるネゴシエーション時に、双方が許容するアルゴリズムから選択される。SSL1.0 と SSL2.0、SSL3.0 というバージョンがある。

・TLS[10] (Transport Layer Security)

TLS はセキュリティーを要求される通信のためのプロトコルである。

TLS プロトコルは､インターネット上の通信セキュリティーを提供する｡このプロトコルは、クライアント／サーバーアプリケーションが盗聴、改ざんはメッセージ偽造を防ぐように設計された方法で通信できるようにする。

TLS の元になったプロトコルが SSL である。TLS1.0 と TLS1.1、TLS1.2 というバージョンがある。

Qt セキュリティーに関するライブラリには SSL3.0 と TLS1.0 をサポートしているので、それを使うことにした。具体的に SSL3.0 を使うか TLS1.0 を使うのはサーバーと統一する必要である。

第6章　　クライアントソフトウェアの設計と開発

本章では、開発したシステムのうち筆者の担当部分について詳細説明する。筆者は図 6-1 中に赤い点線で囲まれた部分の設計及び実装を担当した。

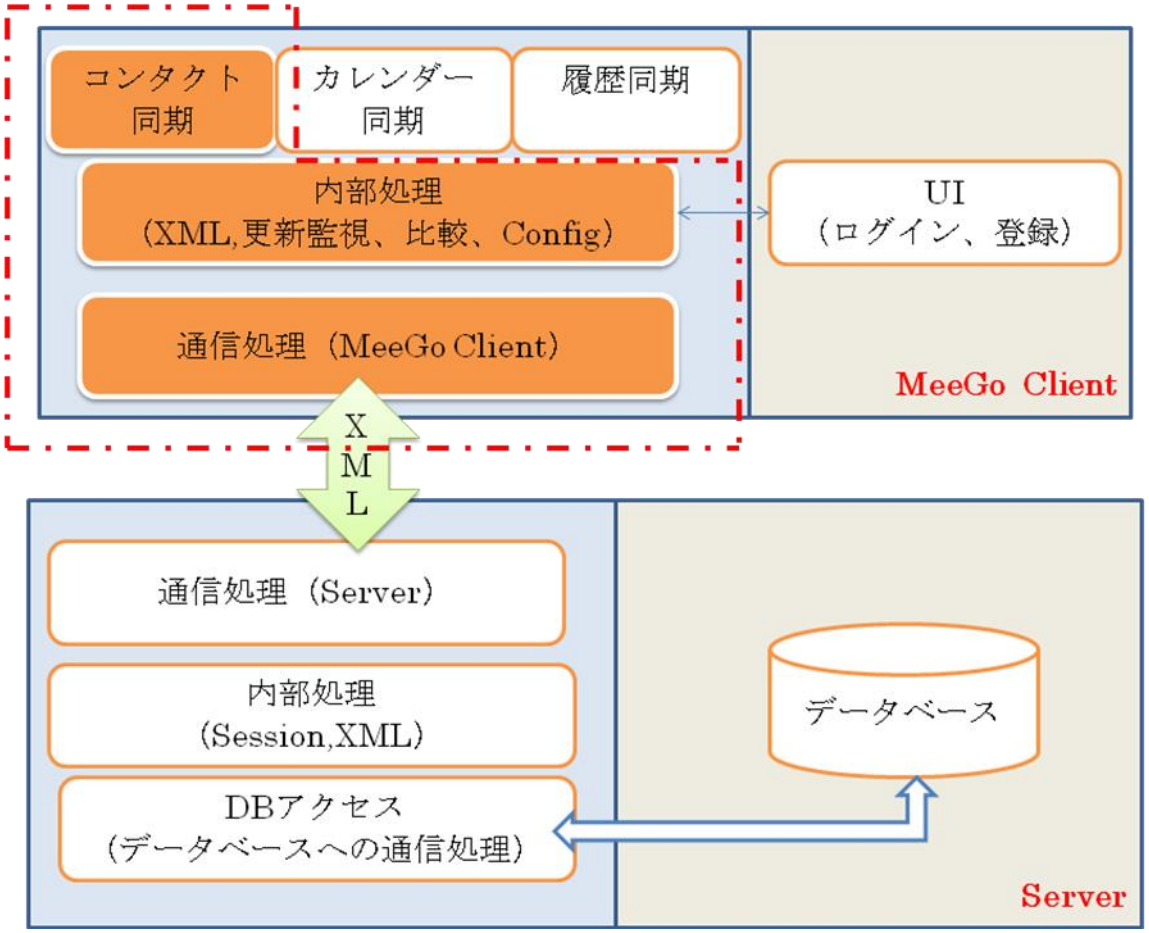


図 6-1　筆者の担当範囲

6.1　　クライアントソフトウェア担当機能の設計

本節では、担当機能の概要、処理の流れ、ソフトウェア設計及び設計ポイントについて述べる。

6.1.1　　機能概要

担当したサブシステムはクライアントソフトウェアのコンタクト同期、内部処理、サーバーとの通信モジュールによって構成されている。担当機能と他の部分のインターフェースを図 6-2 に示す。

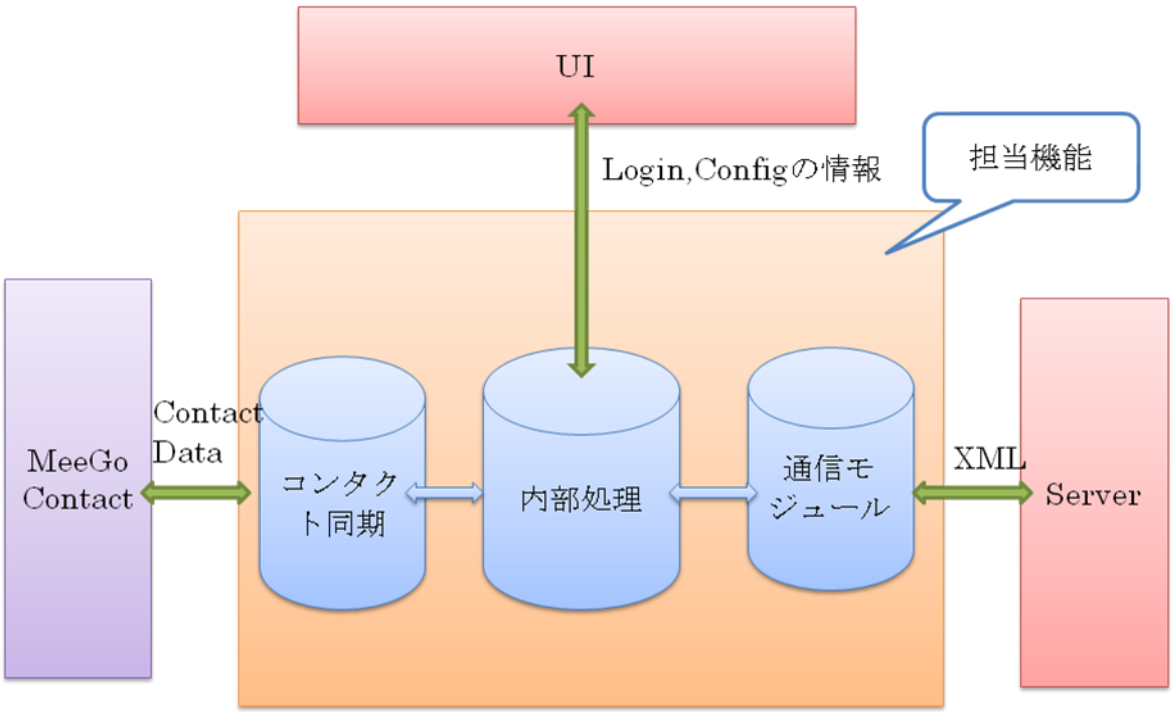


図 6-2　担当部分と他の部分のインターフェース

6.1.2　　機能要件

筆者はクライアントソフトウェアのコンタクト同期、内部処理と通信モジュールの開発を担当した。具体的に以下の機能を実現した。

●コンタクト同期

コンタクトデータの取得や更新を実現する。具体的にコンタクトからデータの取得、コンタクトにデータの追加や変更、削除という操作を提供する。

● 内部処理

クライアントの内部処理は具体的に以下の機能を備えている。

- ✧ Config
  UI 画面から ID とパスワード、設定情報を取得し、設定を行う。また Socket 通信に関するサーバーの IP アドレスやポート番号、MeeGo デバイス ID 基本情報の設定機能も持つ。
- ✧ 更新監視
  MeeGo コンテンツについて、30 秒ごとに監視を行い、新しいデータを生成した場合、更新されたデータを取得する。
- ✧ DBCopy
  比較処理を行うため、サーバーにデータを送信する際に、送られたデータ DBCopyFile.txt に書きこむ。また、サーバーから新しいデータを貰ったら、コンテンツに同期する際に、そのデータを DBCopyFile.txt に更新する。
- ✧ 比較処理
  30 秒毎にコンタクトなどのコンテンツから取得したデータと DBCopyFile.txt から取得したデータを比較し、差分を取る。また、サーバーからコンタクトなどのコンテンツのデータが送られたら、そのデータ取得し、DBCopyFile.txt に更新する。
  更新後の DBCopyFile.txt と MeeGo コンタクトなどのコンテンツのデータを比較し、

差分のデータを取る。

- ✧ XML[11]
  クライアントとサーバーXML をベースにして通信する。サーバーに送信すべきデータを XML に変換する。また、サーバーから受信されたデータについて、XML 解析を行う機能である。

● 通信処理
  クライアントとサーバー間のやり取りはすべて Soket による通信[12]を行う。また、セキュリティを高めるため、暗号化で通信する。

6.1.3 非機能要件と制約条件

本節は、筆者担当した部分の非機能要件と制約条件を述べる。

◆ 非機能要件

- ✧ 使用性
  コンタクト更新データがあったら、30 秒以内に更新したデータをサーバーに送信可能とする。また、サーバーから新しいデータが送られたら、すぐにコンタクトに同期できる。
- ✧ 汎用性
  内部処理と通信処理に関しては、汎用性の高いモジュールを実現する。コンタクトに対応するのみではなく、カレンダーやメールなどのコンテンツ同期機能にも利用される。
- ✧ 保守性
  クラス毎に分けて機能を追加すること、フィードバックを取り入れやすくするため、変更に対応しやすい設計を行う。

◆ 制約条件

MeeGo と Qt フレームワーク[13]を用いて、モジュールを開発したことで、MeeGo デバイスのみに対応する。他のデバイスには利用できない。

6.1.4 通信処理と内部処理の流れ

本節は、サーバーと通信処理の流れとクライアント側の内部処理の流れを述べる。

● サーバーと通信処理の流れ

クライアントの通信処理はまず MeeGo デバイス（端末）はサーバーに接続することで開始される。次にユーザは ID とパスワードを用いて認証を行う。認証が終わり次第データ共有機能は開始される。データ共有機能の通信の流れとしては、まず、クライアント側は MeeGo デバイスのデータが更新されたら、その更新されたデータをサーバーに送信する。次にサーバーはそのデータを受信し、データベースと同期すると共に他のログインしているデバイスに送信する。他のクライアントはサーバーから送られたデータを受信し、MeeGo デバイスに同期する。具体的な処理の流れを図 6-3 に示す。

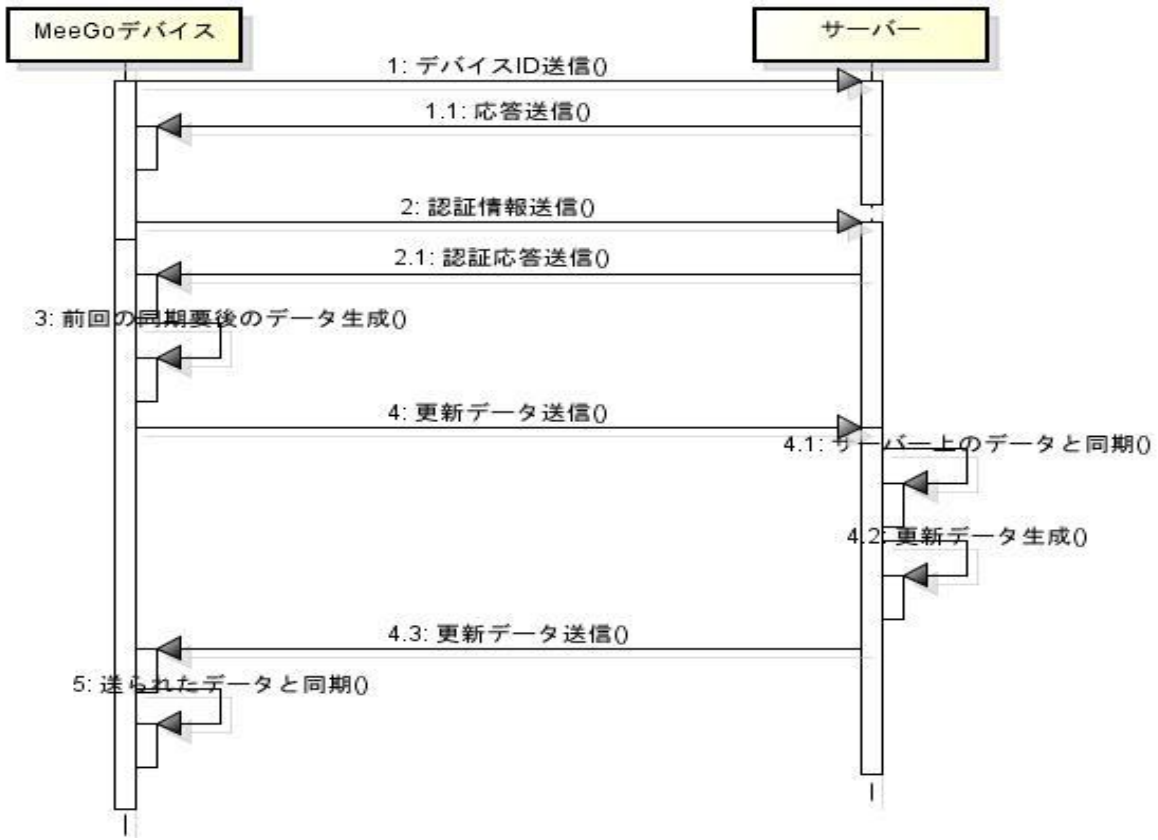


図 6-3　サーバーと通信処理のシーケンス図

● 内部処理の流れ

内部処理の流れを図 6-4 に示す。主に MeeGo デバイスのデータをサーバーに送信するまでの送信とサーバーから送られたデータを MeeGo デバイスに同期するまでという 2 つの流れから構成される。

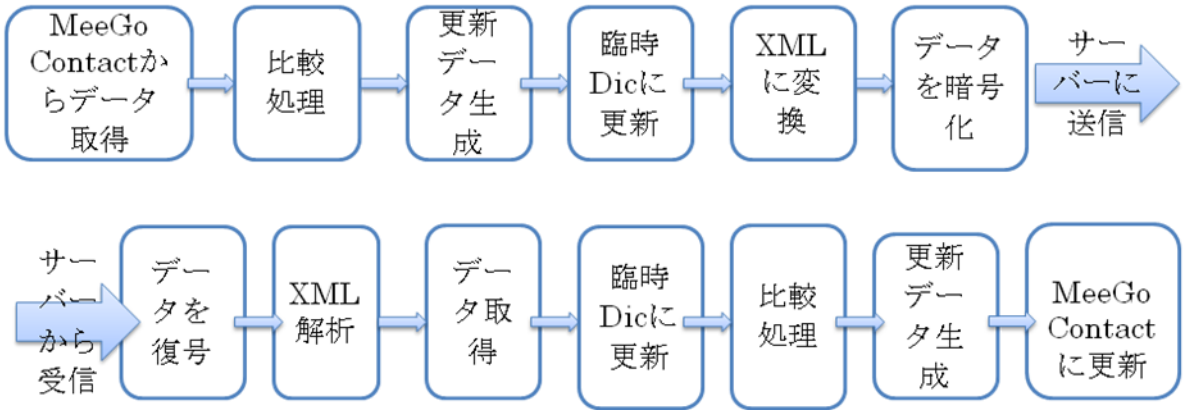


図 6-4　クライアントソフトウェア内部処理の流れ

6.1.5　　開発方針

開発を 2 段階に分けて進めてきた。イテレーション 1 では、中間報告Ⅱまで最小限の機能を実現する。具体的には MeeGo エミュレータ[14]環境でコンタクトの共有機能を完成させる。イテレーション 2 では、内部処理機能の改善、実機への対応を行い、評価を実施する。

6.1.6 ソフトウェア設計

担当した部分の機能要件を基づいて、内部処理と通信処理機能の機能を中心と位置づけ、UML モデル[15]をベースによるソフトウェアの設計を行った。具体的にはまず、ソフトウェアを構成する分析クラス図、各クラス間のやりとりを表すための分析シーケンス図を作成し、次にそれを基に詳細設計クラスを作成した。

クラス構造設計では、ただ機能を実現するだけではなく、再利用性や保守性を高めるように意識して、クラス設計を行った。内部処理は機能が多く、比較と XML モジュール、コンタクト、DBCopyFile4 つクラスが分かれている。通信処理機能は通信デーモンクラスに任せる。また、UI からのデータや DeviceID については設定というクラスに処理されている。コンテンツ同期と DBCopyFile に関する処理はべつべつのクラスに実現される。各クラスの責務は以下のようになる。分析クラス構造を図 6-5 に示す。

- 設定クラス

UI とのインターフェースである。また Socket 通信に関するサーバーの IP アドレスやポート番号、MeeGo デバイス ID などに関する設定を行う。

- 通信デーモン

暗号化でサーバーとの送受信機能である。主に設定した情報やコンタクトの内容をサーバーに送信する。またサーバーから送られたデータを受信する。

- XML モジュール

定めた標準 XML を照らしてサーバーへの送信時にデータを XML に変換する。また送られたデータを XML 解析する。

- 比較モジュール

コンタクトデータと DBCopySaveFile.txt のデータを取得し、比較処理を行い、差分のデータを取る。

- コンタクト

MeeGo　コンタクトからデータを取得し、” Uuid,Content,lastModifiedTime,isValid”の文字列に変換する。また、”Uuid,Content,lastModifiedTime,isValid”という文字列をコンタクトに定めたデータ型に変換し、MeeGo　コンタクトのデータを更新する。

- DBCopyFile

コンタクトのデータをサーバーに送信すると同時に“Uuid＊Content＊lastModifiedTime”という形で ContactCopyFile.txt に書きこむ。また、サーバーから送られたデータにより、コンタクトを同期する際にそのデータを使って、ContactCopyFile.txt を更新する。

- Peoplemodel

Qt のライブラリを使って、コンタクトの追加、削除、変更、取得などの機能を実現する。

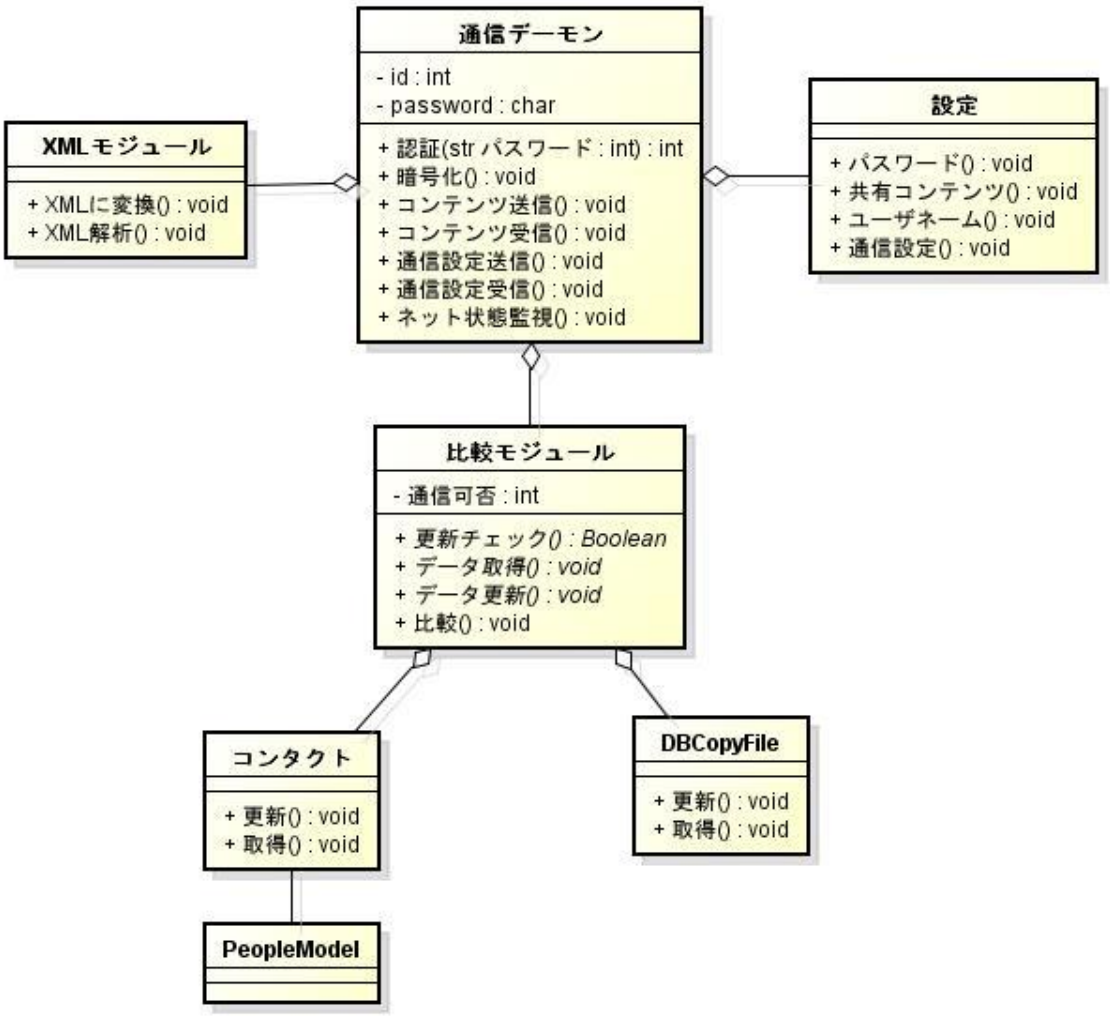


図 6-5　クライアントソフトウェア担当部分の分析クラス図

また、内部処理と通信処理は、設定、コンタクト、DBCopyFile、比較、XML、通信モジュールから構成され、各モジュール間のやり取りを図 6-6 にまとめた。

１．コンタクトは設定から同期設定（同期 ON/OFF の設定）を取得する。
２．通信は設定かログイン情報（ユーザ ID、パスワード、デバイス ID）を取得する。
３．比較はコンタクトからデータを取得する。
４．比較は DBCopyFile からデータを取得する。
５．コンタクトから取得さられたデータと DBCopyFile から取得されたデータを比較し、差分のデータを生成する。
６．生成したデータを XML に変換する。
７．XML 変換したデータを暗号化にして、サーバーに送信する。
８．通信はサーバーから送られたデータを受信し、データを復号する。
９．XML はそのデータを解析する。
１０．　比較は解析されたデータを取得し、またコンタクトおよびDBCopyデータと比較し、差分データを抽出。
１１．　差分データを用いて、MeeGo　コンタクトに更新する。
１２．　差分データを用いて、DBCopyFile に更新する。

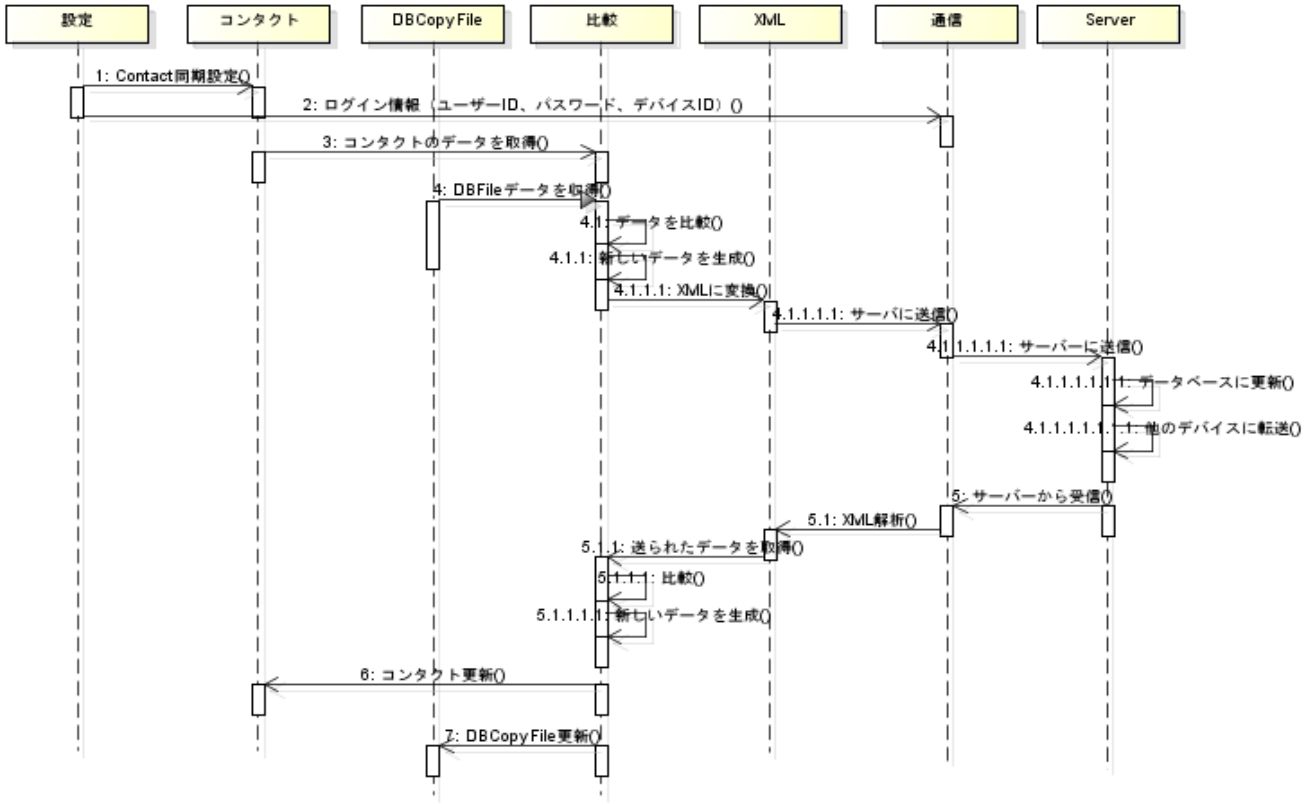


図 6-6　クライアントソフトウェア担当部分の分析シーケンス図

クラス詳細設計では、分析クラス図と分析シーケンス図を基にクラス詳細設計を行った。各クラスの機能詳細は以下のようになる。設計クラス詳細を図 6-7 に示す。

Network クラスは、暗号化による Socket 通信を実現するメインクラスである。

XMLPacket クラスは、Socket 通信に関するコマンドやデータ、コードなどの変換と解析の機能を持つ。

TimeThread クラスは、コンタクトデータと DBCopySaveFile.txt のデータを取得し、比較処理を行い、差分のデータを取る。比較方法としては、同じ Uuid であれば更新時間を比較し、新しい更新されたデータを取得する。

MessageSignal クラスは、サーバーと KeepAlive のため、20 秒毎に起動し、KeeAlive のメッセージをサーバーに送信する。

ContactSignal クラスは、30 秒毎に起動し、コンタクトが更新された場合は、更新されたデータをサーバーに送信する。

Contact クラスは、MeeGo プラットフォーム中に既に存在している「meego-app-contacts」[16]のクラスである PeopleModel クラスと PeopleModelPriv クラスを利用し、コンタクトのデータの取得と更新を行う。

PeopleModel クラスは、MeeGo プラットフォーム中に既に存在している「meego-app-contacts」のクラスである。Qt Contacts[17]に関する API を使って、コンタクトの操作機能を持つ。

PeopleModelPriv クラスは MeeGo プラットフォーム中に既に存在している「meego-app-contacts」のクラスである。コンタクト操作に関するパラメータの定義を行う。

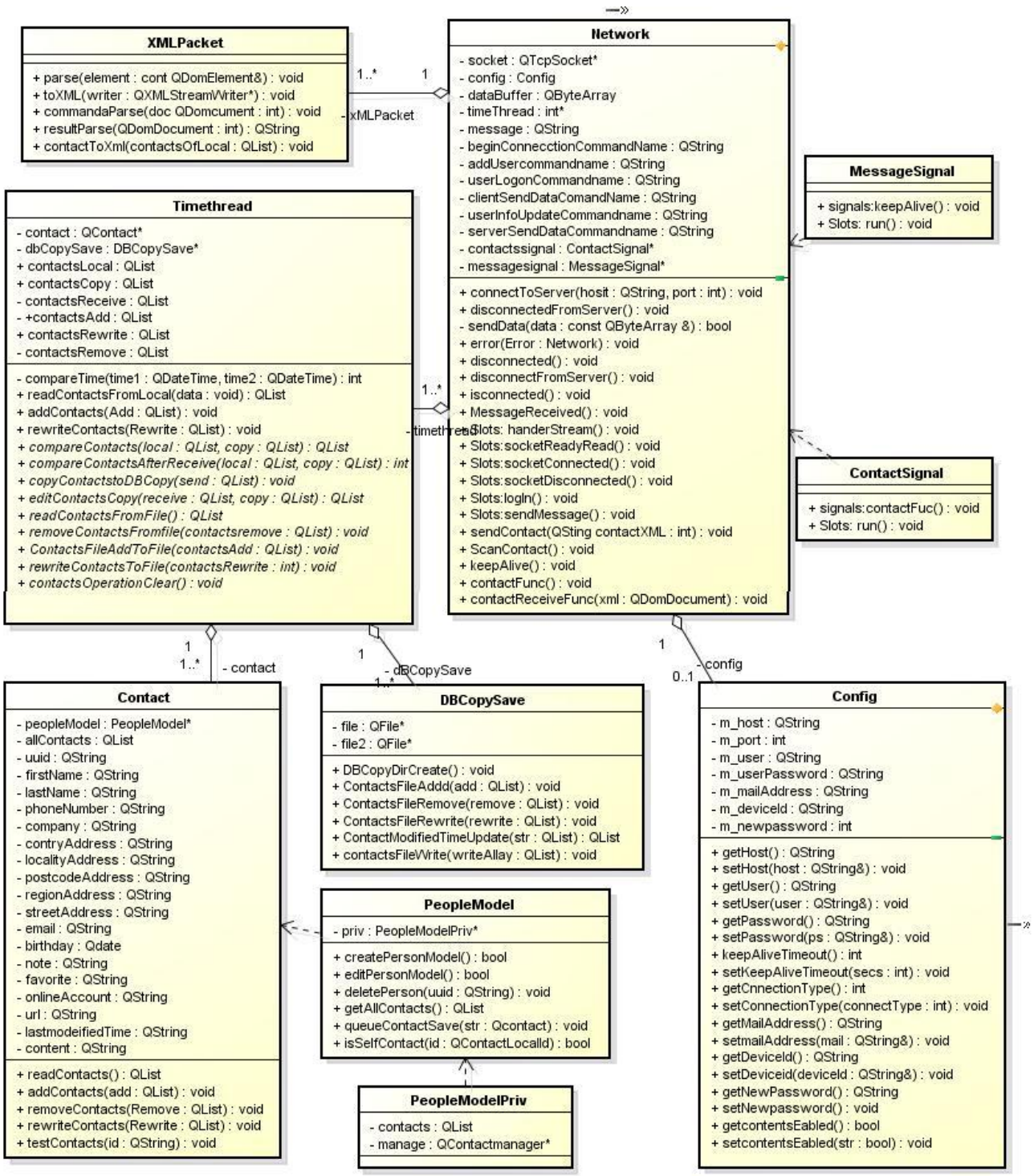


図 6-7　クライアントソフトウェア担当部分の設計クラス図

6.2　　クライアントソフトウェア担当機能の開発

6.2.1　　コンタクト同期モジュール

コンタクト同期モジュールは主にコンタクトデータの取得や更新を実現する。MeeGo 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。

コンタクト同期の設計クラスは Contact クラスと PeopleModel クラス、PeopleModelPriv クラスから構成される。PeopleModel クラスと PeopleModelPriv クラスは MeeGo プラットフォーム中に既に存在している「meego-app-contacts」のクラスである。その２つのクラスを利用する上で、コンタクトクラスを追加し、コンタクト同期機能を実現できた。

- コンタクト同期の実装

イテレーション１においては、エミュレータ環境における実装を行った。
まずはコンタクトデータの追加、削除、更新及び取得の詳細方法を明確するため、コンタクトテストプロジェクトを実装し、テストを詳しく行った。その後に、テストプログラムを参考にして、コンタクトの同期機能を実現した。

イテレーション２では、実機に対応するため、設計と実装の修正を行った。エミュレータで動作されるプログラムは実機で正しく動かなかった。原因は実機のバージョンはエミュレータより新しいためであった。それによって、peoplemodel クラスをそのままで使えなかった。そして LocalID も使えなった。実機に使われている「meego-app-contacts」を参考にして、peoplemodel クラスを修正し、LocalID から Uuid[18]に変更を行い、実機でテストをしながら、コンタクト同期を実現させた。

- コンタクト同期処理の流れ

コンタクト同期処理の流れを図 6-8 に示す。比較処理モジュールからの比較処理の結果を取得して、その結果を基づいて、処理を行う。具体的には以下のように説明する。

１．まず、新しい追加すべきデータがあった場合は、そのデータから Uuid や名前、電話番号、メールなどの詳細情報を取得する。
２．変更すべきデータがあれば、そのデータからから Uuid や名前、電話番号、メールなどの詳細情報を取得する。
３．Qt の QContactSaveReqest クラスを使って、Uuid や名前、電話番号、メールなどの詳細情報をコンタクトに保存する。
４．Qt の QContactSaveReqest クラスを使って、コンタクトマネージャーを更新する。
５．削除必要なデータがあった場合は、削除データから Uuid を取得する。
６．MeCard でなければ、Qt の QContactSaveReqest クラスを使って、その Uuid に関するコンタクトを削除する。
７．Qt の QContactSaveReqest クラスを使って、コンタクトマネージャーを更新する。

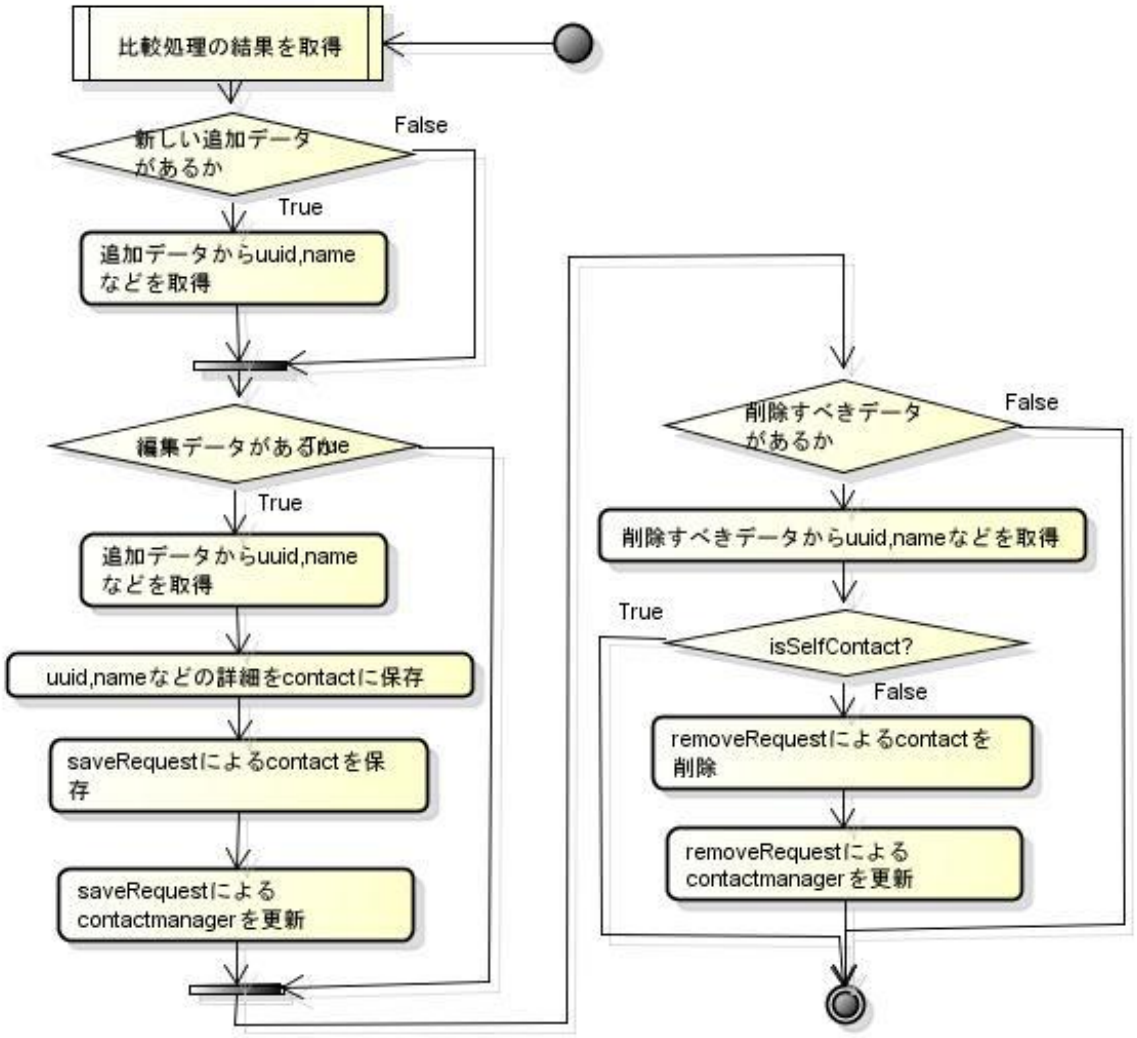


図 6-8　コンタクト同期処理の流れ

6.2.2　　DB モジュール

コンタクトでは､新しいデータを生成すると､新規データをサーバーに送信するとともに、ContactCopyFile を更新する必要がある。また、サーバーから新しいデータが送られた場合は、そのデータを用いて、コンタクト同期するとともに ContactCopyFile を更新する必要である。　DB モジュールは ContactCopyFile の生成及び ContactCopyFile データの取得と更新という機能を持つ。

●DB モジュールの実装

DB モジュールの機能はプログラムにおいて、DBCopySave クラスで実現される。ContactCopyFile 型はテキストファイルである。読み取りやすいため、ContactCopyFile には一つの Contact が一行に保存される。また、追加や削除、変更の操作は読込みと同じように一行ずつで行う｡変更という操作を例として､具体的には処理の流れは以下のようになる。フローチャート図を図 6-9 に示す。

１．他のモジュールで変更すべきデータを取得し、EditContact の文字列に格納する。

２．ContactCopyFile.txt から保存されたデータを取得し､FileContact の文字列に格納する。

３． EditContact と FileContact の Uuid を比較し、同じ場合は、EditContact にその同じ Uuid を持つコンタクトを削除する。
４． EditContact の LastedModifiedTime を現在の時間に更新する。
５． LastedModifiedTime が更新された EditContact を FileContact に追加し、ContactCopyFile.txt に書きこむ[19]。

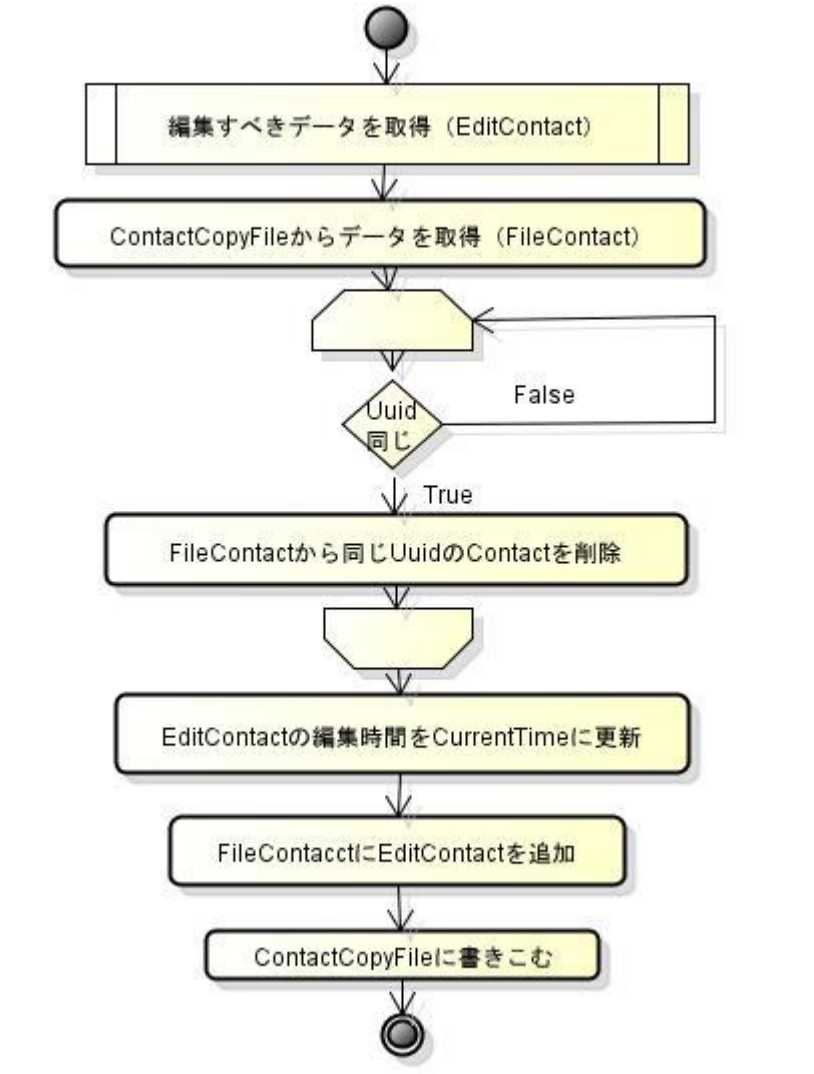


図 6-9　DBCopyFile 操作のフローチャート図(データ変更する例)

6.2.3　　XML モジュール

ログイン、パスワード変更、新規登録、コンタクト同期、Keep alive など内容をすべて XML ベースにしてサーバーと通信する。メッセージの種類に応じて、Command という要素名で定義される。送られた XML メッセージを要素名によって解析し、簡単に取得することができる。XML モジュールにコンタクトデータの XML 標準出力の一つ例を図 6-10 に示す。

- Command
  メッセージの種類を示す。例えば ClientSendData であれば、クライアントからサーバーへデータ送信という属性を特定する。

- Contact
  コンタクトのデータを送られた属性である。「Calentar」なら、カレンダーのデータを送られたと示す。
- Uuid
  一つの操作対象に与えられた意な認証番号である。例えば、MeeGo　コンタクトに対して、一つの連絡先に一つの Uuid が設定される。
- Content
  操作対象の内容を全部でこちらにまとめる。内容の項目は“、”によって、分けられている。
- lastUpdateTime
  操作対象の最新更新時間を示す属性である。またサーバーとクライアントの時間表記は同じで要求される。
- IsValid
  送られたデータは有効かどうかを示す属性である。通常は“0”となる。データが削除され、無効な場合は“1”である。

```
<?xml version="1.0" encoding="UTF-8" ?>
<MeegoSync>
 <Command>ClientSendData</Command>
        <Data>
          <Contact>
            <Uuid> {12345678-1244-3456-123456781234} </Uuid>
            <Content>kyo,senka,Tsukuba of Universitay,08039393939,,,,,,,,,,,,</Content>
            <LastUpdateTime>2011/11/08 11:31:25 </LastUpdateTime>
            <IsValid>0</IsValid>
          </Contact>
          </Data>
</MeegoSync>
```

図 6-10　クライアントからサーバーへの XML 出力の一つ例

クライアント側の仕様と開発言語によらず、本システムの標準の XML 文字列が送られて来るとサーバー側はそれを解析し、処理の結果も XML 文字列に変換され、クライアント側に返信するので、クライアント側はその返信を解析し、自分から送信されたリクエストの処理結果を得ることができる。

統一インターフェースによって、サーバー側とクライアント側間のやり取りの流れは標準化される。

6.2.4　　比較処理モジュール

◆同期処理[20]

コンタクト共有機能は 2 つ以上の MeeGo デバイスにある同じコンタクトで同じ内容になるようにする処理である。ある場所にあるファイルに何らかの変更を加えたとき、同期処理によって別の場所にある同じコンタクトにも同じ変更がなされる。

同期処理の方法としてはコンタクト全体をコピーするのではなく、2 つの場所の差分を比較し、差分のみをやり取りして同期を行う。今回はサーバーを経由してデバイス間のデータを共有することで、同期処理は以下の要件を満たす。

① あるデバイスのデータが更新されると、サーバーを経由して更新されたデータのみを他のデバイスに転送する。

② 2 つのデバイスのデータが更新されている場合、更新時間を比較し新しいデータを他のデータを上書きする。古いデータを消してしまう。

③ 比較処理について、ファイルの全体更新時間を比較するのではなく、比較粒度はできるだけ細かくする。例えばコンタクトであれば、一人ずつのデータを比較する。それによって、送信デバイスと受信デバイスの内容が多くの部分で一致する場合、同期のために転送が必要となるデータ量は少なくなる。

◆比較処理の実装

比較処理はサーバーと同期前の比較処理とコンタクトと同期前の比較処理を分けて実装を実現した。まず、サーバーと同期前の比較処理を具体的に説明する。比較処理の流れを図 6-11 に示す。

1．他のクラスで取得されたコンタクトのデータと DBCopyFile のデータを用いて、Uuid と更新時間によって、コンタクト 1 つずつを比較し、データの差分を取る。

2．Uuid 同じで且つコンタクトデータの更新時間は新しい場合は、その Uuid のデータを新しい更新されたデータとしてサーバーに送る。

3．コンタクトデータにおいて、新しい Uuid のデータがあったら、その Uuid のデータを新しい追加されたデータとしてサーバーに送る。

4．DBCopyFile データにおいて、新しい Uuid のデータがあったら、その Uuid のデータをコンタクトより削除されており、削除されたデータとしてサーバーに送る。

また、コンタクトと同期前の比較処理を具体的に説明する。

1．サーバーから送られたデータとコンタクトのデータを用いて、Uuid と更新時間、IsValid によって、比較し、データの差分を取る。

2．Uuid 同じで且つ IsValid が” 1”でしたら、サーバーから送られたデータは他のデバイスから削除された無効なデータとみられ、removeList に追加する。

3．Uuido 同じで、且つ IsValid が” 0”で,更新時間を比較し、サーバーから送られたデータの更新時間は当たらい場合、その Uuid を持つデータは他のデバイスから更新され、editList に追加する。

4．サーバーから送られたデータにおいて、新しい Uuid を持ち、IsValdid が” 0”である場合、その Uuid を持つデータは他のデバイスから追加され、addList に追加する。

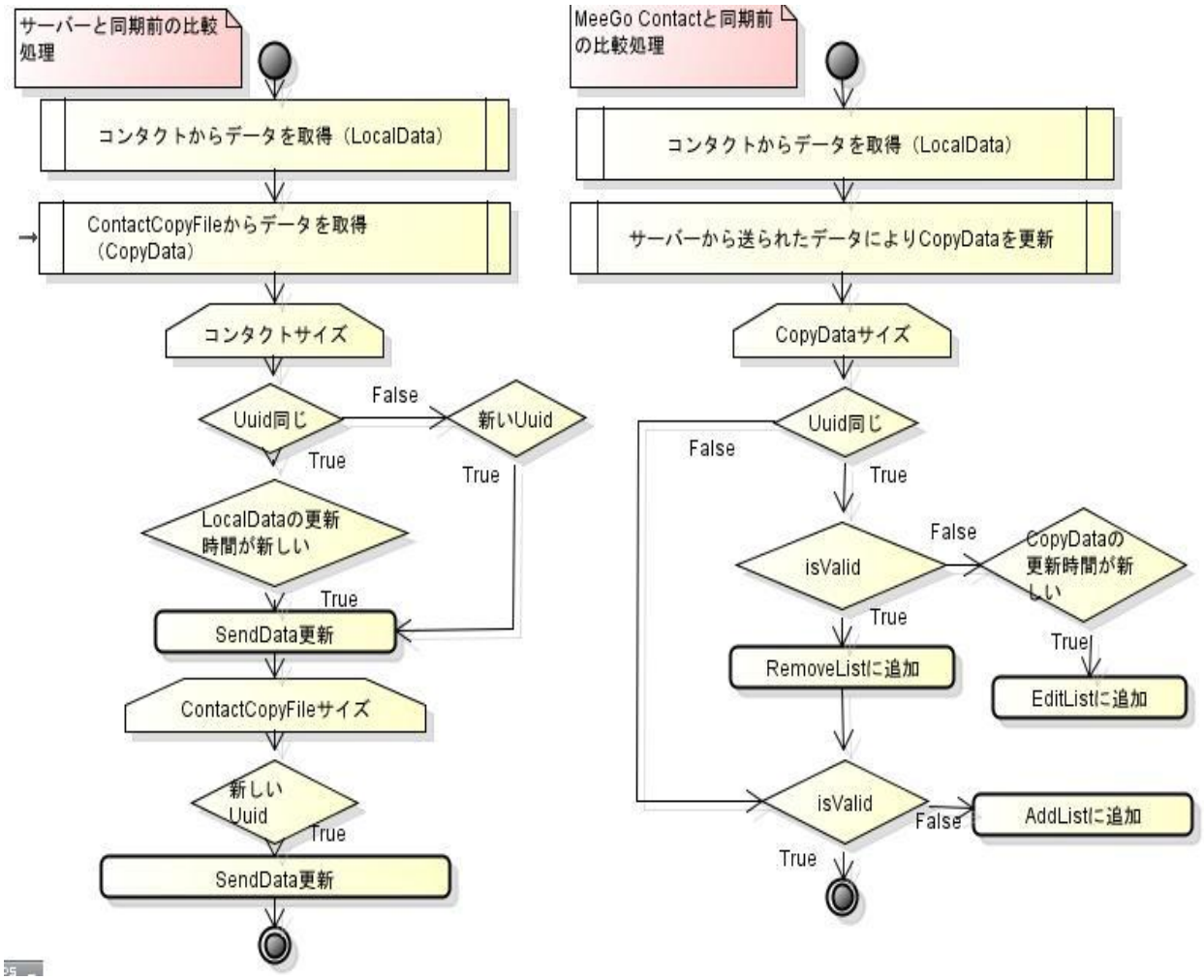


図 6-11　比較モジュールのフローチャート図

6.2.5　　サーバーと通信処理モジュール

通信処理モジュールについて、筆者はイテレーション 1 フェーズまで開発を行った。主に QTcpSocket に関する API を使って、基本機能を実装した。使用するアドレスは IP　version4 のアドレスである。通信プロトコルは TCP、UDP の二種類あるが、今回情報量はすくなくても、特に送信ミスが発生してしまう場合、MeeGo デバイス間でデータ同期することができなくなるため、TCP を用いることに決定した。

◆ソケット通信における通信の流れ

通信処理機能の開発の前提知識として、ソケット通信における通信の流れを説明する。ソケット通信のモデルはクライアントサーバーモデルとなっており、サーバーはクライアントからの接続要求を待ち、クライアントからの接続要求があると、それを受けて通信が接続される。通信接続後はクライアント、サーバーの一方が受信待ちの状態にある場合に、相手から何らかの要求が送信された時に、これを受け付けることができる。これをどちらかが通信を切断するまで行われる。

ソケット通信における具体的な通信の処理の流れを図 6-12 に示す。

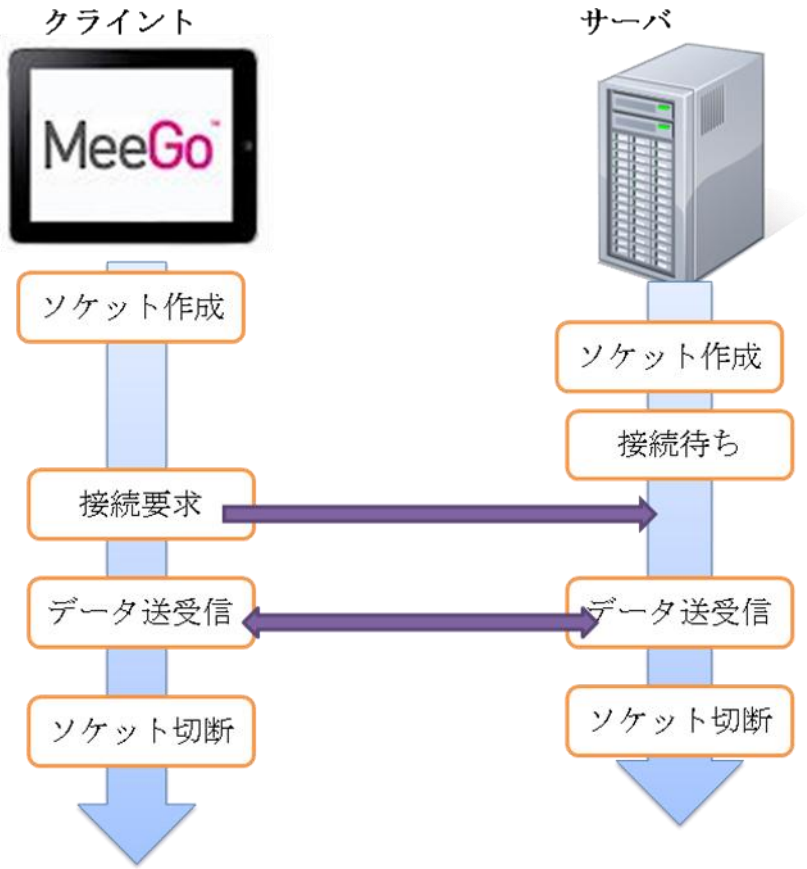


図 6-12　ソケット通信処理の流れ

◆シグナルとスロットを用いた非同期通信の実装

signal/slot[21]のメカニズムは Qt の最も中心的な特徴である。シグナルはある特定のイベントが起こった時に発信される。Qt のウィジットはすでに定義されている多くのスロットをもっているが、通常はサブクラスを作成して自分自身で定義したシグナルを追加する。スロットは特定のシグナルに反応して実行される関数である。スロットも同じようにサブクラスを作成して自分自身で追加することによって、反応したいシグナルだけを扱うことができる。

今回は Qt の QTcpSocket クラスが提供する非同期モードの処理を用いて、非同期通信を実現させた。具体的に signal/slot を用いて、非同期モードの処理を実装した。シグナルとスロットによるオブジェクト間通信を図 6-13 に示す。シグナルとスロットの通信は Qt の connect メソードを利用することで実現された。

例えば、Socket クラスの connected は、最後に実行された時点での Socket の接続状態を取得する。Network クラスは handleStream を呼び出して、socket の接続の現在の状態を確認する。

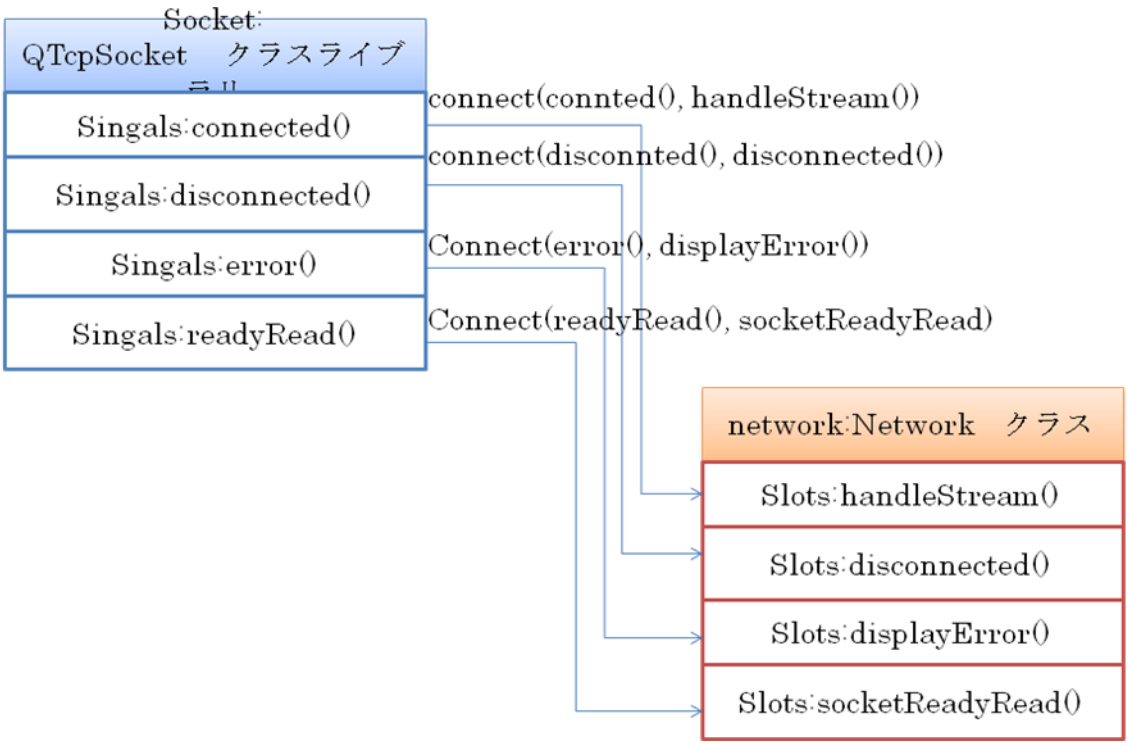


図 6-13　シグナルとスロットによるオブジェクト間通信

◆クライアントソケット通信モジュールの設計および基本機能の実装

筆者はイテレーション１において、通信モジュールの設計及び基本機能の実装を行った。ソケット通信に関する基本機能は Network クラスに実現された。具体的な処理の流れを図 6-14 に示す。

１．クライアントソケットを起動し、初期化を行う。
２．設定されたホストやポート番号を使って、サーバーに接続する。
３．サーバーと接続したら、サーバーへ MeeGo デバイス ID を送信する。
４．デバイス ID を送信した後にユーザの操作によって、通信処理を行う。具体的な通信処理部分の実装はチームメンバの劉建宇が担当する。

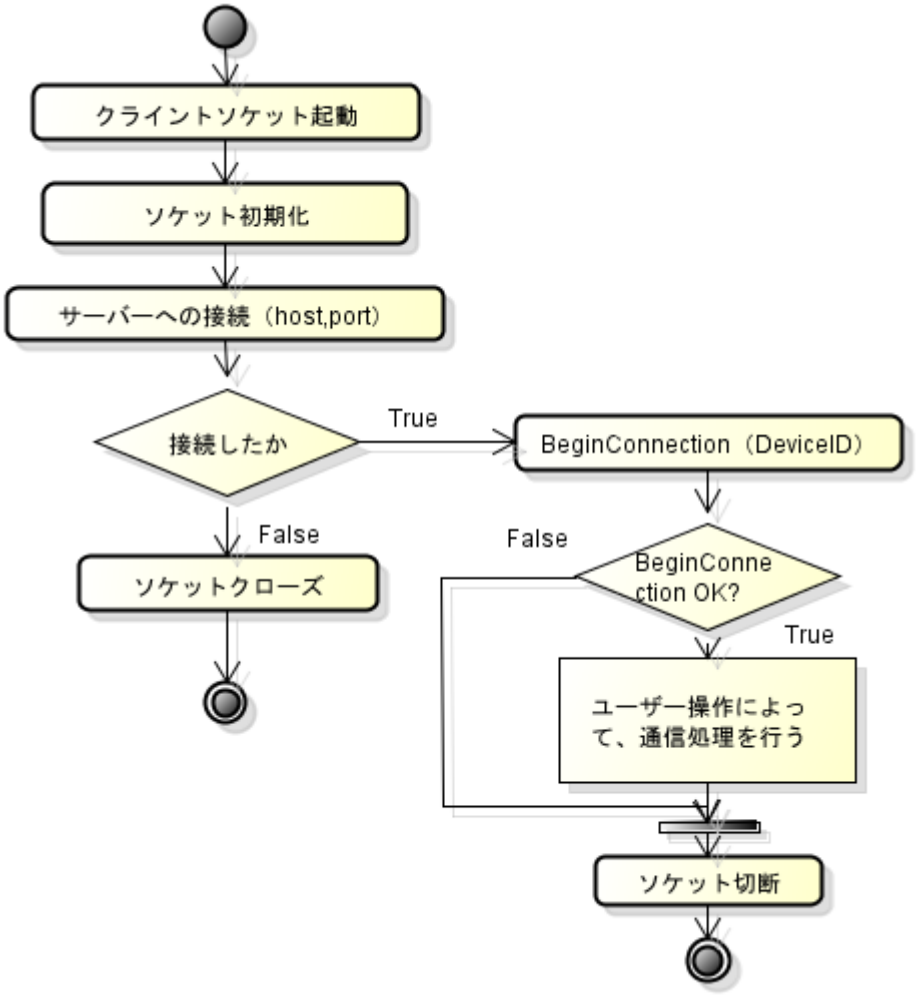


図 6-14　クライアントソケット通信のフローチャート図

6.2.6　　設定モジュール

クライアントソフトウェアの内部処理、通信モジュールと UI（本システムログイン・新規登録画面）のインターフェース部分である。また Socket 通信に関するサーバーの IP アドレスやポート番号、MeeGo デバイス ID 基本情報の設定機能も持つ。設定モジュールは他の部分とのインターフェースを図 6-15 にまとめた。本システムログイン・新規登録画面にてユーザが入力されたデータまたは MeeGo デバイス取得されたデータを設定して、通信モジュールはそのデータを取得することができる。

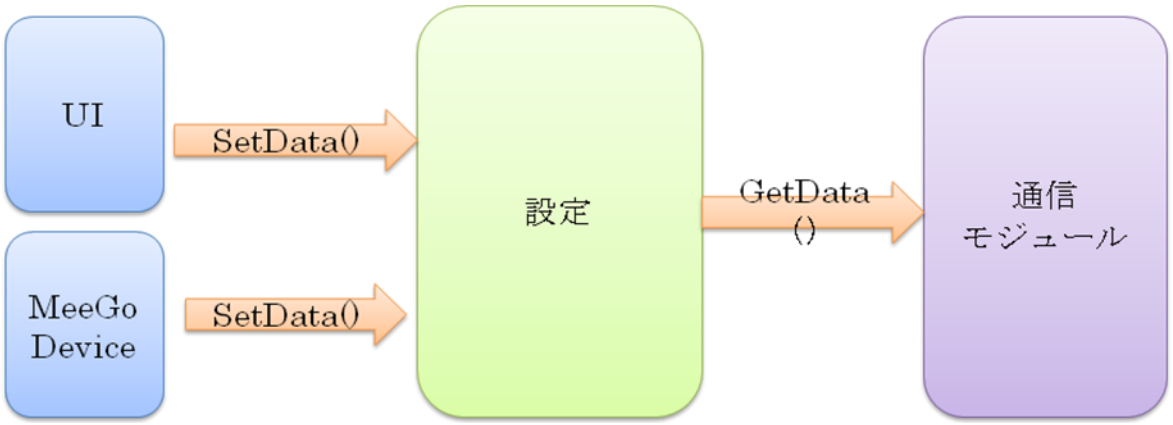


図 6-15　設定モジュールと他の部分のインターフェース図

◆設定モジュールの設計及び実装

一般的にシステムを利用する際にユーザはログイン画面より登録する必要があるが、デバイス ID は要らない。本システムユーザ ID とパスワードを除く、デバイス ID も必要がある。本システムはひとつのユーザは複数の MeeGo デバイスを利用することで、サーバー側はどちらのデバイスでデータ更新されたかどうかを認識するため、デバイス ID をサーバーと接続する際にサーバーに送信する必要である。デバイス ID は画面から設計すれば操作性がよくないため、プログラムで直接にデバイスの無線 LAN のハッドウェアアドレスを取得して、デバイス ID として使われる。Config クラスにそれらの設定機能を実現した。新しい設定があれば、すぐにこのクラスに追加することができるので、拡張性や利用性の高い設計を実現した。

6.3　開発実績

本節では、本プロジェクトにおける筆者は担当した機能開発についての実績を述べる。

6.3.1　ドキュメントについて

クライアントソフトウェア（コンタクト同期、内部処理、通信モジュール）の開発において、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、概要クラス及び説明をチーム共同で作成した。詳細設計フェーズでは、クライアントソフトウェアの詳細設計書、詳細クラス及び説明を作成した。実装では、クライアントソフトウェアのコーディング基準書を作成した。テストでは、コンタクト同期、内部処理及び通信モジュールの単体仕様書、クライアントとサーバープログラムの結合テスト仕様書を作成した。評価フェーズでは、コンタクト同期および通信機能の評価表を作成した。

6.3.2　プログラムについて

実装したプログラムの実績を表 6-1 に示す。

表 6-1　実装したプログラムファイル表

成果物	機能名	ファイル名	行数

C++ ソースコード	設定	config.h config.cpp	221 ステップ
	通信処理	network.h network.cpp messagesignal.h messagesignal.cpp contactsignal.h contctsignal.cpp	1151 ステップ
	XML モジュール	xmlpacket.h xmlpacket.cpp	145 ステップ
	比較処理	timethread.h timethread.cpp	607 ステップ
	DB モジュール	dbcopysave.h dbcopysave.cpp	364 ステップ
	コンタクト同期	contact.h contact.cpp peoplemodel.h peoplemodel_p.h peoplemodel.cpp	548 ステップ

第7章　クライアントソフトウェア担当機能の評価

本章では、開発担当したクライアントソフトウェアの通信処理、コンタクト同期機能の評価について、評価方法、評価結果及び考察について述べる

7.1　評価方法

評価試験では、MeeGo タブレット実機を用いた評価試験を行う。実機の仕様を表 7-1 に示す。試験の目的は開発担当した部分について、要件定義書にて定めた機能と非機能要件が達成可能か検証する。

試験方法では、サーバーやテストプログラムを利用して、通信処理、コンタクト同期処理の結果及び所要時間を計測した。Socket 通信試験は無線通信でサーバーと接続して、
評価項目に書いた内容に沿って、ひとつずつ試験を行った。コンタクト同期試験では、テストプログラムを作成し、コンタクトデータの取得、追加、変更及び削除の機能をテストした。また、サーバーに送られた 10 件、50 件コンタクトを受信してから同期までかかる時間を調べた。試験では、10 件コンタクトの追加、変更、削除は 300 回まで試験を行い、追加処理の時間、変更処理の時間及び削除処理の時間を検証した。また 50 件コンタクトの追加、変更、削除は 100 回まで試験を行い、追加処理の時間、変更処理の時間及び削除処理の時間を検証した。

表 7-1　MeeGo　タブレット実機の仕様

	仕様
実機モデル	ONKYO　TW317
OS	meego-tablet-ia32-pinetrail-1.2.0.90.7.20110706.4
CPU	インテル Atom
ハードディスク	32GB
ディスプレイ	マルチタッチ/ハイビジョン対応 11.6 型ワイド液晶
無線通信機能	IEEE802.11b/g/n
バッテリー駆動時間	約 5 時間
軽さ	約 1.0kg

7.2　評価結果

7.2.1　機能評価結果について

担当したクライアントソフトウェアの Socket 通信とコンタクト同期の機能評価結果を表 7-2 に示す。

表 7-2　機能評価項目と結果

評価内容		結果
大分類	細分	
	サーバーとの接続	ログインや新規登録の時にサーバー

Socket 通信		と接続できた
	20 秒毎に KeepAlive メッセージをサーバーに送信	20 秒毎に KeepAlive メッセージを送った
	ユーザログオン時にサーバーに送信	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できた
	ユーザ登録時に送信	新規登録の時にユーザ ID とパスワードをサーバーに送信できた
	ユーザパスワード変更時に送信	パスワード変更の時にユーザ ID とパスワード及び新しいパスワードをサーバーに送信できた
	コンタクトをサーバーに送信	コンタクト更新を検出した場合、自動的にサーバー送信できた
	サーバーから送られたコンタクトを受信	サーバーから送られたコンタクトを受信できた
	ノートをサーバーに送信	コンタクト更新を検出した場合、自動的にサーバー送信できた
	サーバーから送られたノートを受信	サーバーから送られたコンタクトを受信できた
コンタクト同期	新しいコンタクトに追加	1 件から 4000 件までのコンタクトを追加できた
	指定されたコンタクトを削除	Uuid による指定されたコンタクトを削除できた
	指定されたコンタクトのデータを変更	Uuid による指定されたコンタクトを変更できた
	30 秒ごとにコンタクトのデータを取得	全てコンタクトの取得ができた

7.2.2 性能評価結果について

１件から 300 件まで連続でひとつコンタクトをサーバーから受信して追加するまでかかる時間の結果について、平均時間、最大時間と最小時間が表 7-3 に示す。また分散データを図 7-1 に示す。

表 7-3　１件から 300 件まで一つコンタクトを追加する時間

項目	時間(ms)
平均	114
最大	266
最小	8

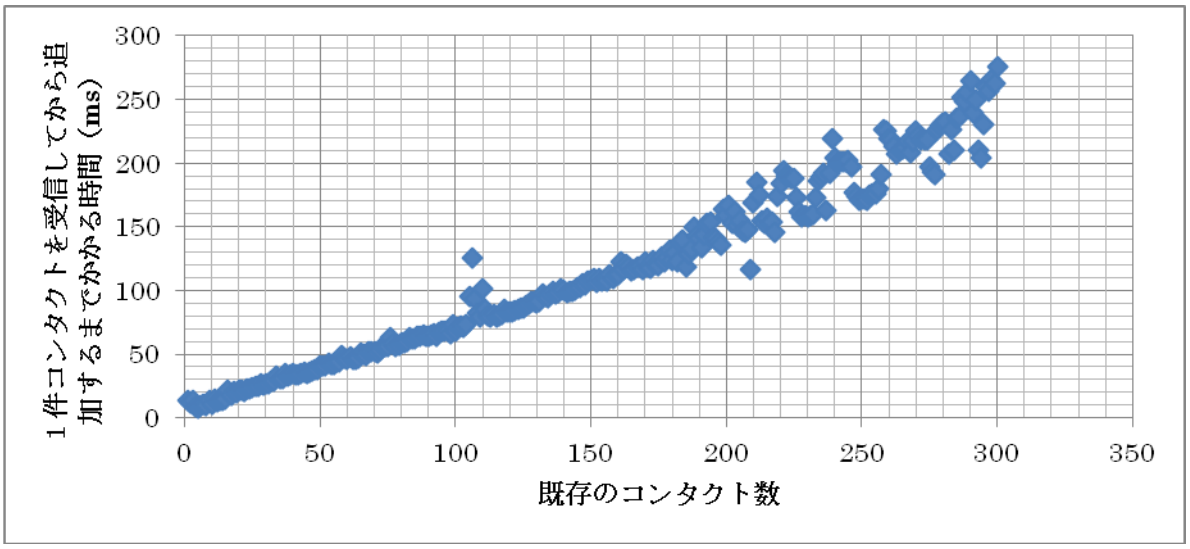


図 7-1　1 件から 300 件まで一つコンタクトを追加する時間

10 件コンタクトを受信してから追加、変更、削除の試験を 300 回まで繰り返しテストを行った。その結果については、平均時間、最大時間と最少時間を表 7-4 に示す。追加、変更、削除ごとの分散データを図 7-2、図 7-3 及び図 7-4 に示す。

表 7-4　10 件コンタクトを受信してから同期する時間

項目	追加(ms)	変更(ms)	削除(ms)
平均	274	487	384
最大	343	628	551
最小	204	396	283

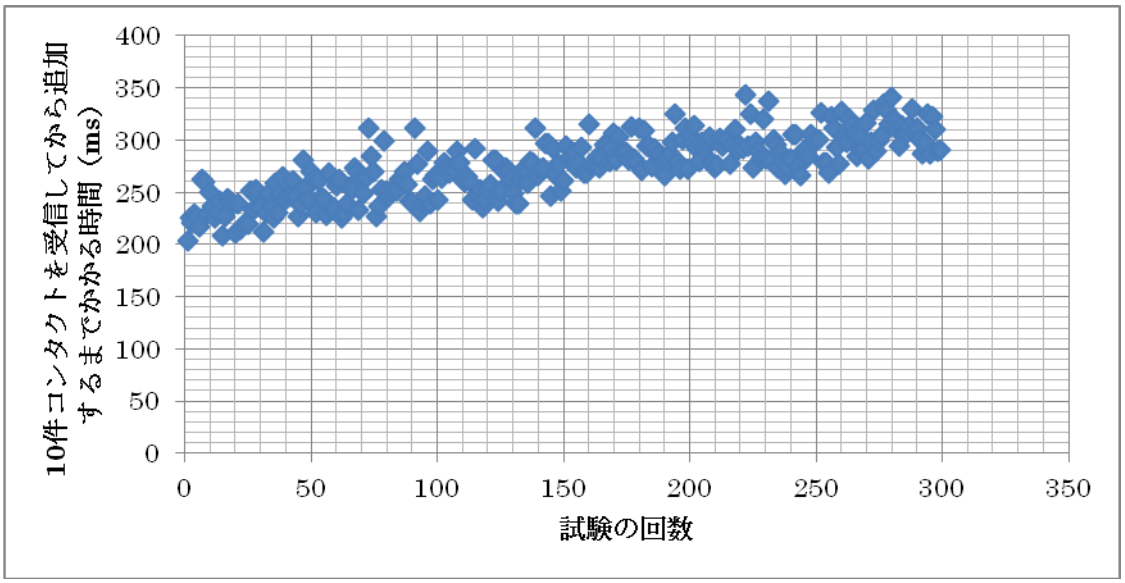


図 7-2　10 件コンタクトを受信してから追加するまで時間

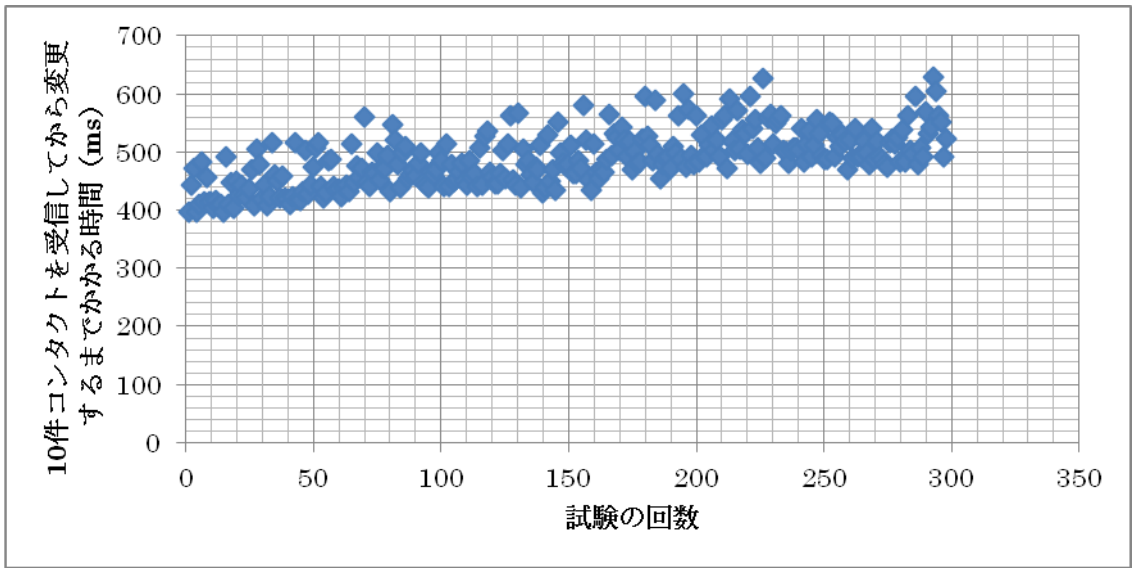


図 7-3　10 件コンタクトを受信してから変更するまで時間

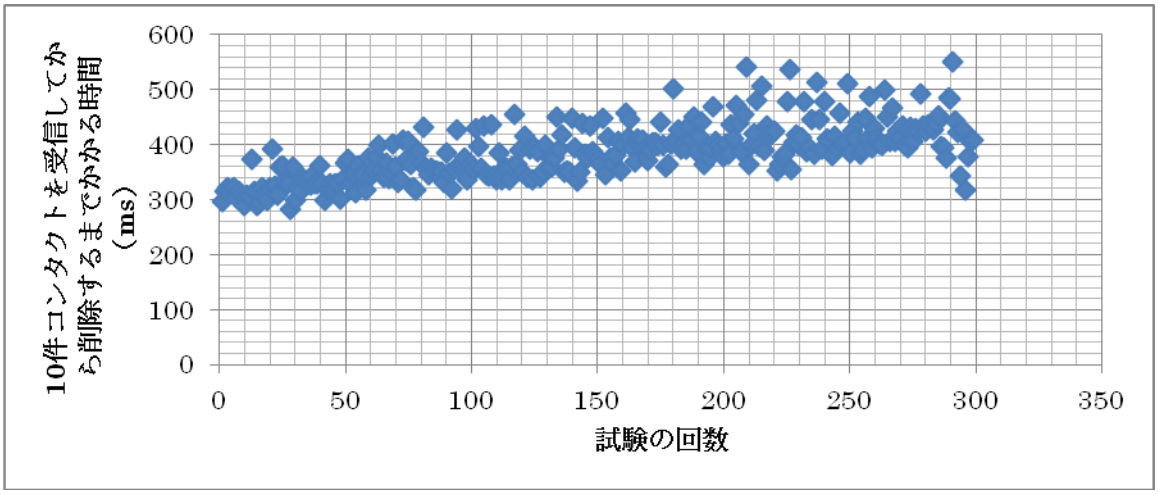


図 7-4　　10 件コンタクトを受信してから削除するまで時間

50 件コンタクトを受信してから追加、変更、削除の試験を 100 回まで繰り返しテストを行った。その結果については、平均時間、最大時間と最少時間を表 7-5 に示す。追加、変更、削除ごとの分散データを図 7-6、図 7-7、図 7-8 示す。

表 7-5　50 件コンタクトを受信してから同期する時間

項目	追加（ms）	変更（ms）	削除（ms）
平均	1729	5388	3066
最大	2028	6277	3339
最小	1571	4882	2702

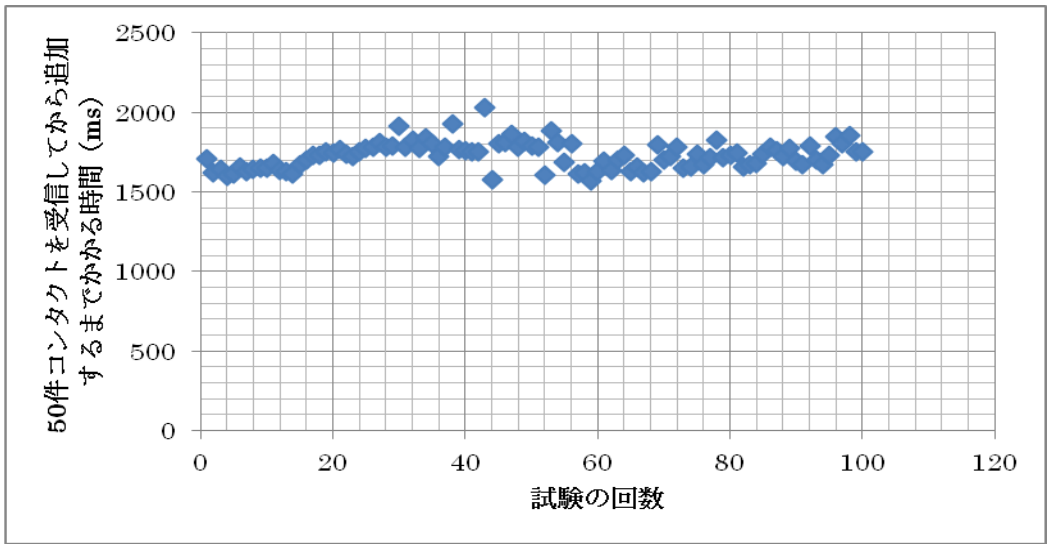


図 7-5　50 件コンタクトを受信してから追加する時間

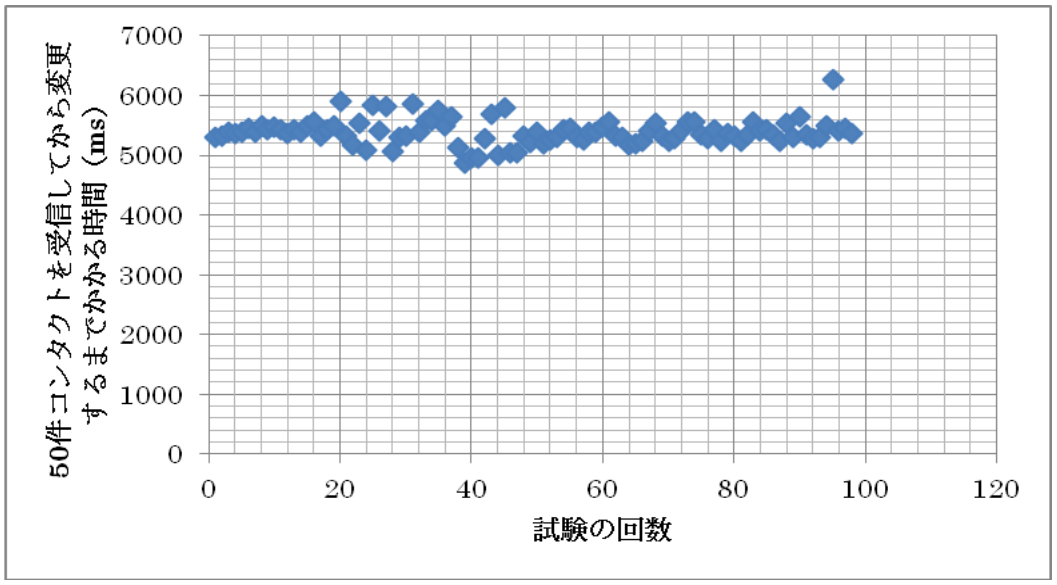


図 7-6　50 件コンタクトを受信してから変更する時間

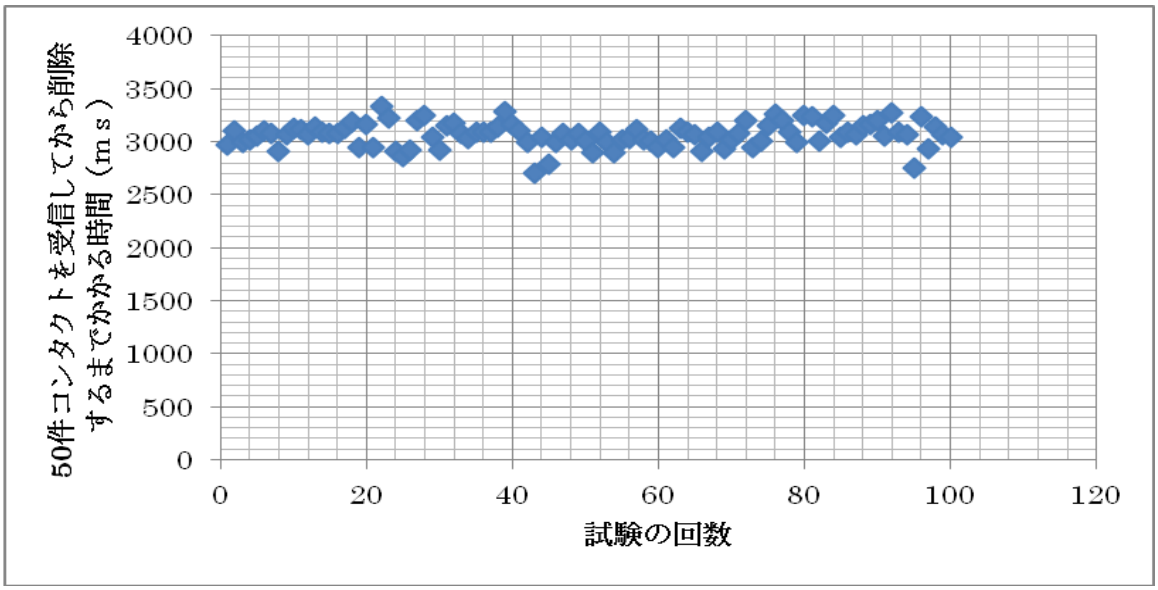


図 7-7　50 件コンタクトを受信してから削除する時間

7.3　　考察

評価試験の結果により、要求定義書に定めた機能と非機能要件が達成したと考えている。

Socket 通信機能は、試験結果により、クライアントプログラムを起動すると、自動的に UI 画面で新規登録やログイン及パスワード変更の情報をサーバーに送信することができた。またコンタクトのデータが更新されると、更新されたデータをサーバーに送信することができた。　逆にサーバーから送られた更新データを受信することができた。まだ、サーバーの IP アドレスとポート番号は UI 画面で設定するのではなく、クライアントプログラムで直接設置されたため、サーバーの IP アドレスが変わる場合、そのままクライアントソフトウェアはサーバーと接続できない。今後の課題ではサーバーの IP アドレスとポート番号が UI 画面で設定できるようにする。

コンタクト同期機能では、試験結果により、1 件コンタクトから 4000 件までコンタクトを追加したことができた。次に全てコンタクトのデータを取得することができた。そして、指定された Uuid によって、任意コンタクトの変更や削除することができた。

性能については、コンタクトの追加処理は既存のコンタクト数によって処理時間が決められた。既存のコンタクト数は多ければ多くほど、時間が長くなる。それはコンタクトに関する処理の中に XML 解析や比較、コンタクト操作などについて、多くのループが使われているため、多くの時間がかかった。今後の課題としては、コンタクト同期処理のアルゴリズムを改善し、コンタクト同期時間を短くにする。また処理時間のバラつきについては、既存のコンタクト数は多くなると、バラつきが発生しやすい。それは前のコンタクト追加処理を終わっていない、且つ次のコンタクトを追加する処理が来たら、２つのコンタクトを一つ追加リストになる時に、前に追加するコンタクトの処理時間が通常より長い。後ろのコンタクトを追加するかかる時間が通常おり短いと考えている。

10 件コンタクトと 50 件コンタクトを受信してから同期するまでの時間について、追加処理の時間と変更処理の時間、削除処理の時間を比較すると、変更処理の時間が一番長い。それは変更処理を行う時に、変更すべくデータを検索する及びそのデータを編集する時間必要があるので、他の処理より時間が長くなった。また追加と変更、削除毎の処理時間の分散デ

ータにより、バラつきの程度は多くないと考えている。

第8章　プロジェクト進行の工夫と分析

8.1　工夫

プロジェクトにおいて、チームや個人において工夫した点を記述している。

8.1.1　プロジェクトマネジメントについて

プロジェクトマネジメントについて、以下の点に工夫した。

・開発フェーズ１において、週２回ベースで定例会議を行うことで、チームメンバーの進捗や予定を共有した。

・グーグルドキュメント、DropBox を利用することで、会議議事録や成果物を共有した。

・表を用いて、作業予定、作業中、作業完了をメンバー内で把握できるようにした。

・東京に住んでいるチームメンバがいるので、遠隔ミーテイングを活用した。特に実装段階では、色々な問題を起こし、遠隔ミーテイングを活用することで、それらの問題をチームメンバ内でリアルタイムに検討し、効率的に問題解決を図った。

・メンバの役割は明確し、仕事の分担ができていた。そして、タスクを偏らないようにイテレーション 2 で、各メンバの役割を調整し、再明確にした。

・要件定義、設計段階では、成果物を必要最低限に抑えることで、効率的な開発ができた。

・クライアント側はテスト駆動開発を実施した。特にエミュレータ環境や実機でのコンテツのデータ取得、更新について、最初に必要最低限のテストプログラムを実装し、繰り返しテストをした。OK なれば、そのテストプログラムを基に、クラスの設計やクラスの内部構造も考えて、詳細設計や実行を行った。

8.1.2　技術について

技術について、以下の点に工夫した。

・システム設計を行う前に、データ同期の関連技術をしっかり調査した。また、調査した技術をそのまま流用するだけではなく、その実現方法を参考にして、システム設計を行った。

・機能を実現するだけのではなく、操作性や拡張性、セキュリティーなども工夫し、要件定義を行った。

・クライアントとサーバーの開発環境が違うので、サーバー側とクライアント側は緊密的にテストや確認を行いながら、実装を進めてきたため、より効率的に開発ができた。

・サーバー側とクライアント側を分けて開発しており、標準 XML ベースによる通信を行うことで、MeeGo 以外のデバイスにおいても、サーバー側のソフトウェアをそのまま利用し、クライアント側のソフトだけを簡単に修正してデータ共有機能を実現できる。

8.2　分析

本プロジェクト進行における問題点を分析し、今後の活動に改善していく。

8.2.1　チーム活動における分析

１．リスク管理の実施について

プロジェクトの進行中に発生した問題とリスク、その対策案を考えたが、あまり機能しなかった。理由としてはリスクを管理する責任者が決められなかったのではないかと考えている。リスク管理と監視プロジェクトの進行に関する仕事を作業の一部として、担当者を決めるべきだったと考えている。

２．企画立案について

企画立案フェーズでは､MeeGo の特徴である同じソフトを複数のデバイスで利用出来るという点を基に、案を出した。また新規性や機能性、法律面、開発面、評価という点で検討し、「データ共有システム」に絞り込んだ。振り返りみると、新規性と開発面の検討が不足したと気がした。新規性と開発面をもっと検討すれば、もっとよいシステムを作られるのではないかと考えている。

8.2.2 担当した部分開発における分析

クラス設計においては、XML 処理、比較処理、コンタクト処理、DBCopyFile 処理をモジュール毎にわけられているが、各モジュール中に共通部分とコンテンツによって変動部分をわけていないため、新しいコンテンツを追加すれば、変動部分と各コンテンツ共通部分がわかりにくいと気がした。各モジュールにおいて、変動部分と共通部分を分けて、クラス設計を行えば、新しいコンテンツを追加する際に、共通部分はそのまま利用し、変動部分のみを修正することで、追加は簡単に実現できるではないかと考えている。

第9章　まとめ

本報告書では、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における特定課題研究「研究開発プロジェクト」の最終報告書として、日本インテルが提示した「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」というテーマの元に MeeGo デバイスをベースにしたアプリケーション企画立案、開発したシステムの概要、筆者の担当箇所であるクライアントソフトウェアのコンタクト同期機能、内部処理及び通信処理について、報告した。

開発したシステムはクライアント側とサーバー側の２つ部分を振り分けて実施されており、クライアント側はコンタクトなどの MeeGo コンテンツ同期機能を開発した。サーバー側とクライアント側は標準 XML による通信行うことで、MeeGo ではないデバイス間のデータ共有しても、サーバー側はそのまま利用でき、クライアントソフトウェアだけ修正することで、対応できる。なかでも筆者は、クライアント側のコンタクト同期、内部処理と通信処理の機能を担当した。

開発したシステムは、システムテストの結果要件に沿ったものであり、本プロジェクトの目的であるインテル株式会社ビジョンの一つ「ンピューティング・コンテニュアムを実現し、ユーザがデバイスを意識せずにシームレスに活動を行うためのソフトウェアを開発すること」を考えた MeeGo オーペンソフトウェアプラットフォーム上で動作するアプリケーションの開発を達成できたと考えている。

プロジェクト自体は企画立案、要件定義、設計、実装、テスト、評価までおおむね順調に遂行できた。MeeGo プラットフォームに事前学習が不足であったため、詳細設計や実装のところに結果苦労して、スケジュールよりやや遅れてしまった。遅れた部分に対し、チームメンバと協力しながら、ようやく最終的には遅れを取り戻すことができた。

本プロジェクトの実施にあたり MeeGo プラットフォーム、Qt、オープンソースソフトウェアの開発、モバイル組込みデバイスの開発、オブジェクト指向設計など多くの知識と経験を得る事ができた。

今後の課題は、MeeGo のタブレットだけではなく、MeeGo のハンドセット、ネットブックなどの MeeGo デバイスにも対応する。また、データ共有システムでは、MeeGo プラットフォームの制限を超え、他モバイルプラットフォームにも対応することが挙げられる。

本システム開発において、発見した問題点や改善点をインテルに反映し、新しい OS 開発に役に立つことを期待したい。

謝辞

尹栄植様、池井満様はじめとする日本インテル株式会社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､勉強会の開催や細部にわたるご指導を頂きました。誠に有難うございました。

指導教員である田中二郎教授には、大学院 2 年間にわたり大変お世話になりました。本プロジェクトに関する大変多くのご指導およびご支援をいただきました。深く感謝致します。

本プロジェクトの課題担当教員の追川修一准教授には、丁寧且つ熱心なご指導や貴重な意見をいただきました。心より感謝致します。

本報告書執筆にあたり、ご助言を頂くとともに本報告書の細部にわたりご指導頂いた高度 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担当である山戸昭三教授に感謝致します。

また、本プロジェクトのチームメンバである劉宏超氏、 劉建宇氏、 森本悠矢氏からも多くの力、アドバイスをいただきました。共にプロジェクトを遂行できたことに心より感謝致します。

最後に, 大学院 2 年間の留学生活を様々な面で支えてくださった家族､友人など全ての方々に心より感謝致します。

参考文献

[1]Linus Foundation,About,meego.com,http://meego.jp/about/index.html.

[2] Qt-Cross-platform application and UI framework,http://qt.nokia.com.

[3]MeeGo デバイス,https://meego.com/devices/handset.

[4] QML,http://developer.qt.nokia.com/doc/qt-4.8/qdeclarativeelements.html.

[5] .NET Framework,http://msdn.microsoft.com/ja-jp/netframework/ee624167.

[6] Buteo Sync Solution, http://wiki.meego.com/Buteo/SyncML.

[7]syncEvolution(synchronizing personal information management data), DEVELOPMENT, http://syncevolution.org/development.

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

[9] Transport Layer Security Working Group ,Alan O. Freier,hilip Karlton ,Paul C. Kocher, 『The SSL Protocol Version 3.0』November 18, 1996.

[10] T.Dierks Independent E. Rescorla RTFM, Inc, 『The Transport Layer Security (TLS) Protocol Version 1.2』August 2008.

[11]W3C Recommendation 16 August 2006, edited in place 29 September 2006,Extensible Markup Language (XML) 1.1 (Second Edition) .

[12]QTcpSocket Class Reference, http://developer.qt.nokia.com/doc/qt-4.8/QTcpSocket.html.

[13] Jasmin Blanchette 著,Mark Summerfiedld 著, 杵渕 聡 (翻訳), 杉田 研治 (翻訳) , 『入門 Qt４　プログラミング』（大型本）.

[14]MeeGo エミュレータ上からのネットワーク接続方法に http://meego-users.jp/node/201.

[15]グラディ・ブーチ著,オージス総研オブジェクト技術ソリューション事業部訳, 『UML ユーザーガイド』（Version　1.3）.

[16]meego-app-contacts,http://meego.gitorious.org/meego-ux/meego-app-contacts.

[17] QtContacts,http://doc.qt.nokia.com/qt-mobility-snapshot/qtcontacts.html.

[18]QUuid Class Reference,http://developer.qt.nokia.com/doc/qt-4.8/quuid.html.

[19]第 6 章-読み込みと保存,http://elisp.net/qt-doc-ja_JP/tutorials-addressbook-part6.html.

[20]Rsync version 3.0.9 ,released September 23th, 2011, http://rsync.samba.org/.

[21] 第 3 章 SIGNAL/SLOT を使ってみ よう

http://www.usamimi.info/~guiprog/qt_online/chapter3/chapter3.pdf#search='signal/slot'.

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。

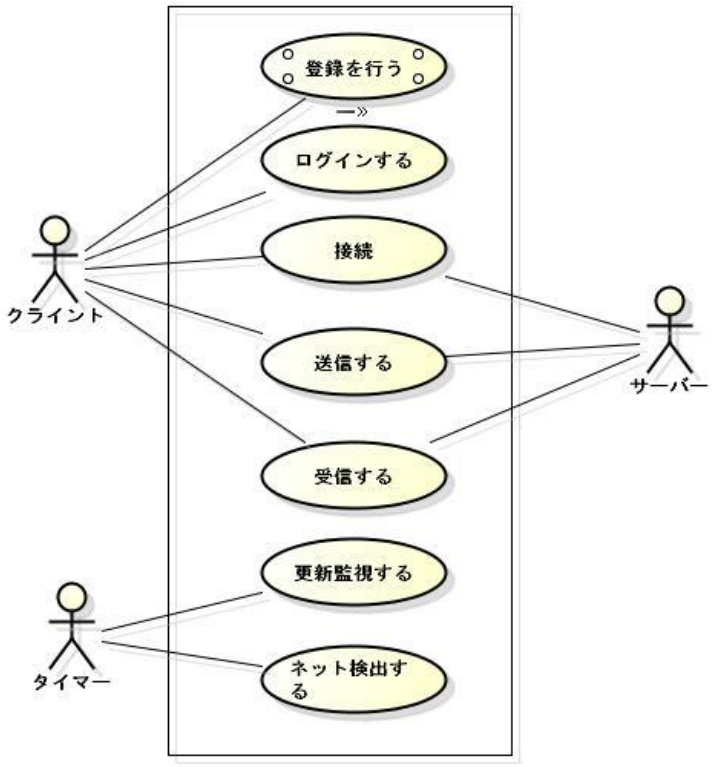
また、法的な制約として、第三者に著作権が存在するメディアファイルを複数のユーザがアクセス可能なサーバ上にアップロードすると、公衆送信権の侵害となる可能性が高い。よって、音楽、動画などのメディアファイルはサーバへのアップロードは行わない。

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

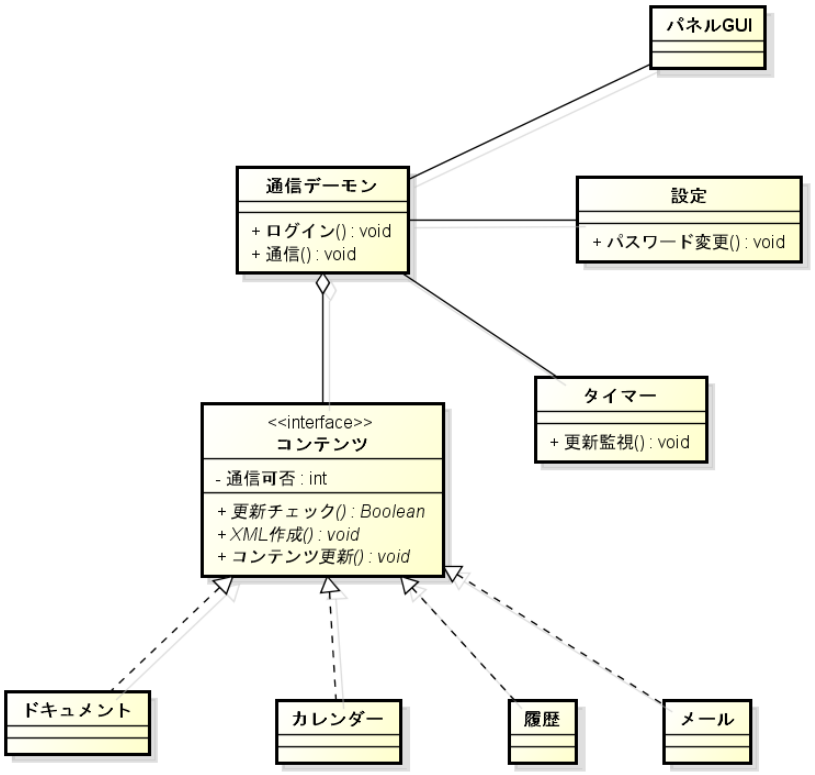
ID	文書名	文書番号	発行年月	備考
	要件定義書			

2.3　定義（用語、略語）

ID	用語・略号	正式表記	意味

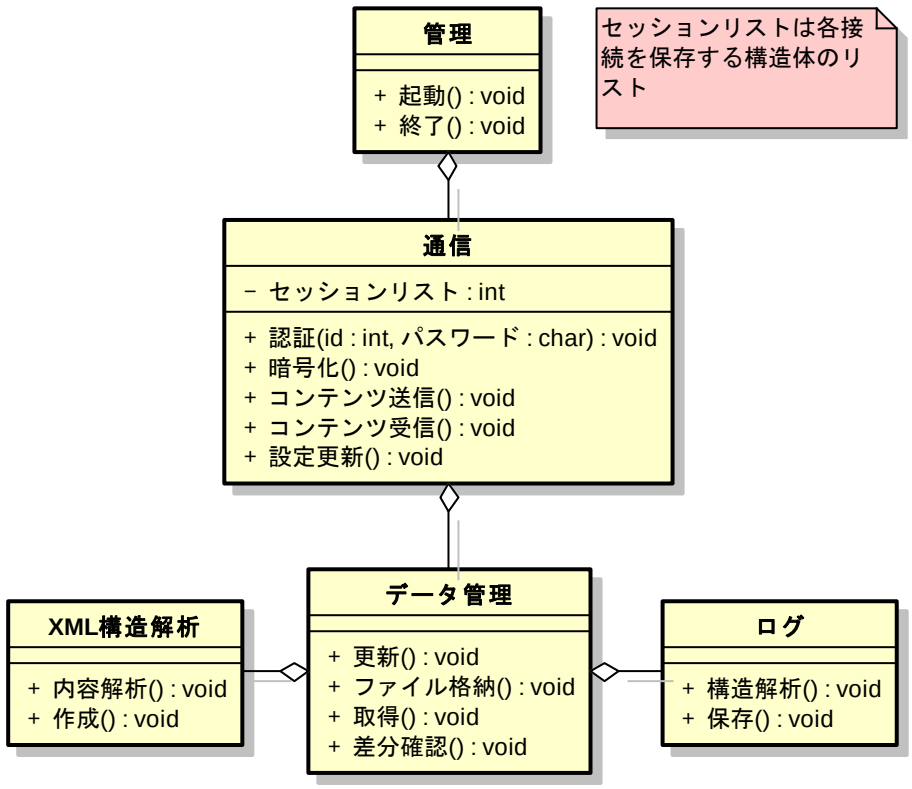
3　システム構成

クライアントソフトウェア



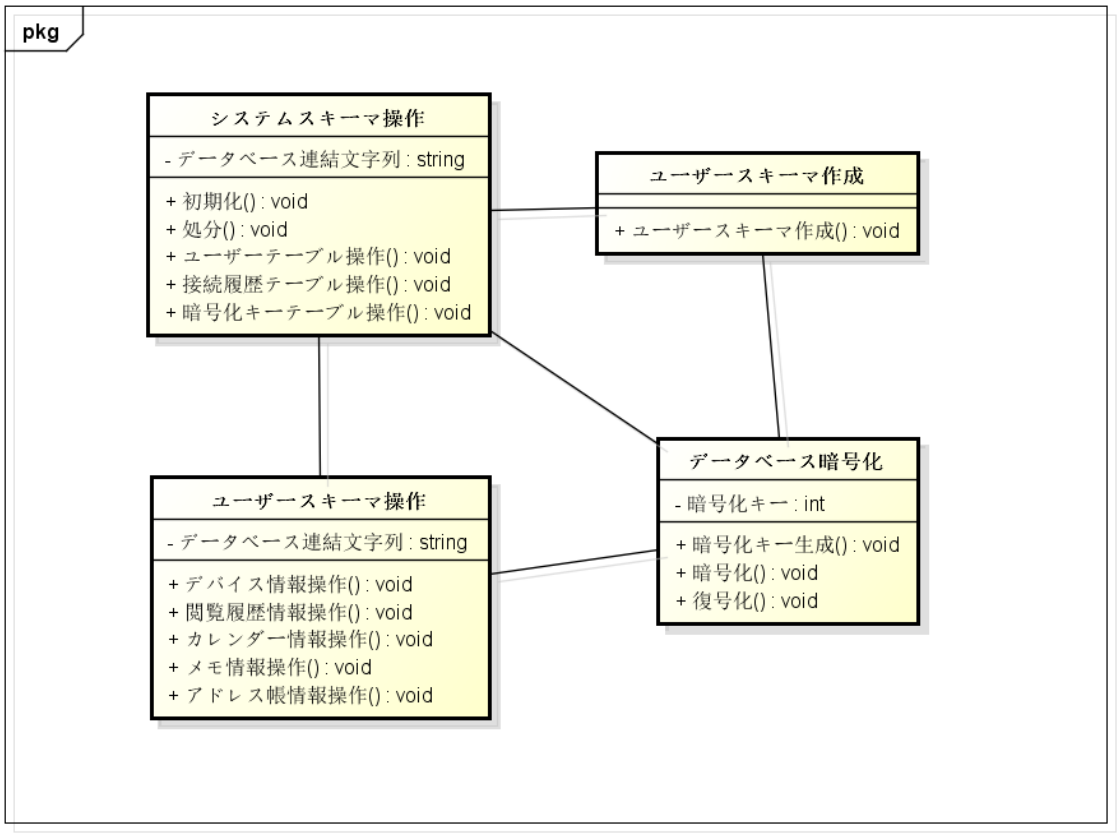
クライアントソフトウェアの分析クラス図を示す。クライアントソフトウェアは画面から入力を受け取り、サーバへの通信を開始する。また、ユーザからの入力によりパスワードの変更などの設定を行うことも可能とする。通信を開始した後はタイマーモジュールで 30 秒周期で各データが更新されているかどうかをチェックし、それぞれの共有項目ごとに XML の作成を行う。また通信デーモンは接続後はサーバからの通信を待ち、データを受信した場合は各項目に更新を行わせる。

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	

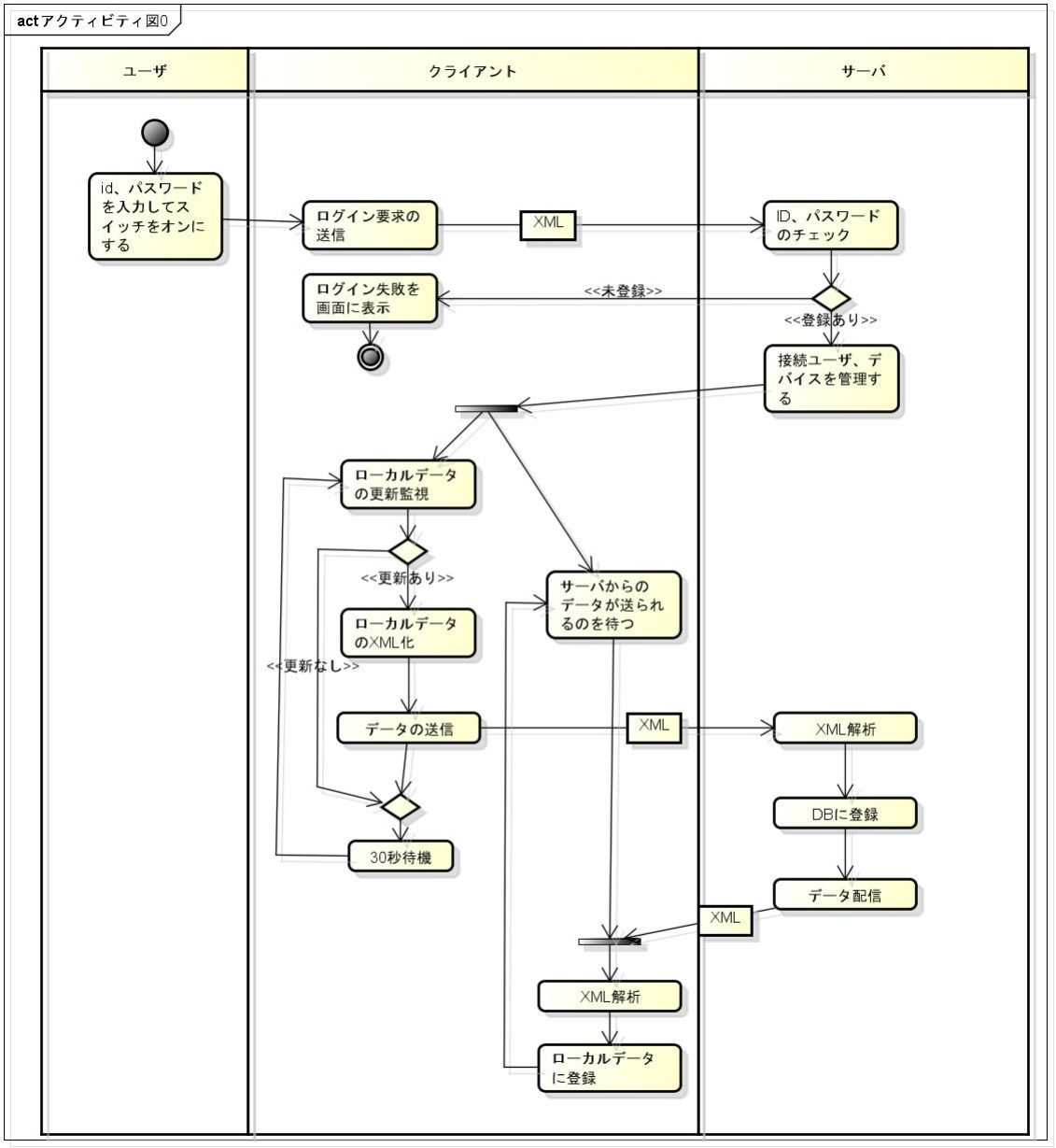
サーバーソフトウェア

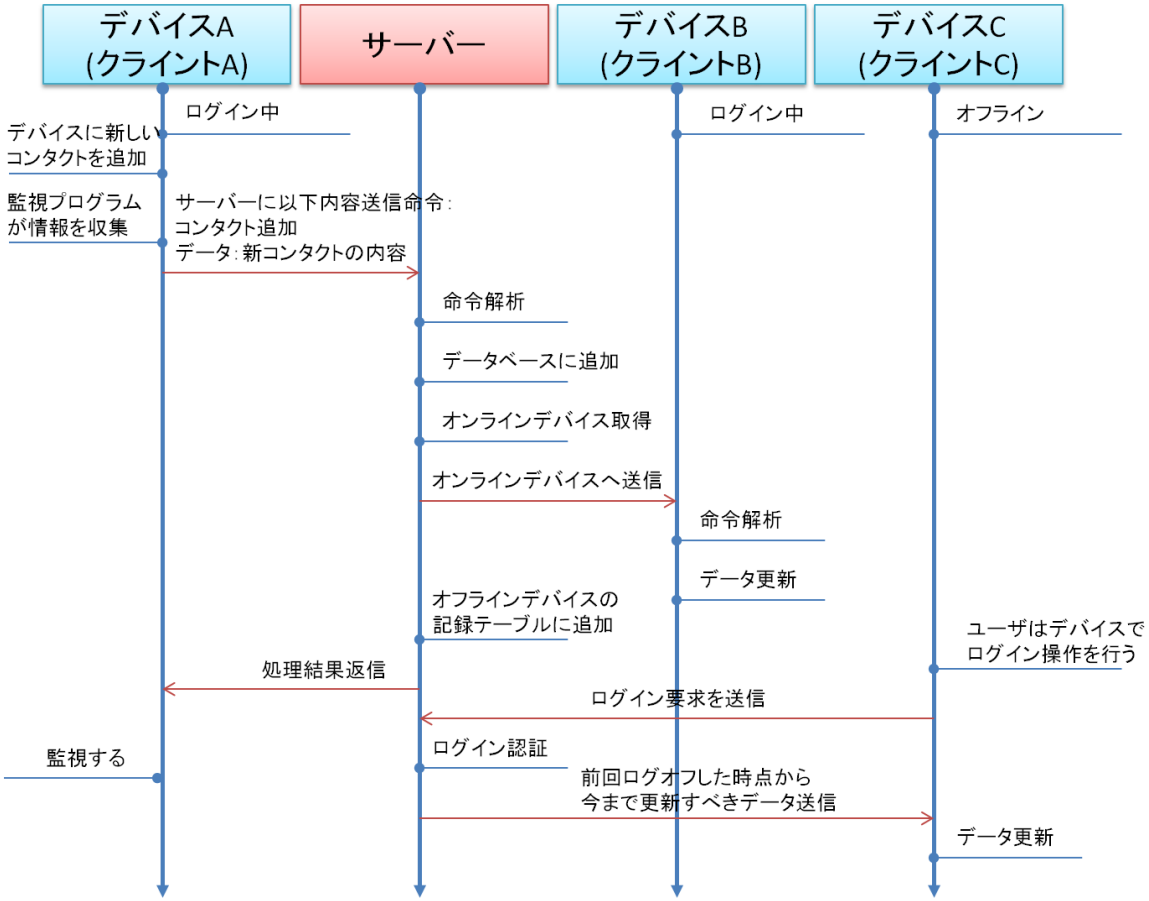
機能 ID	機能名	機能概要	役割分担	関連機能 ID
	通信機能	データの送受信の制御を行う	劉宏	
	接続管理	クライアントの接続を管理する	劉宏	
	XML 生成・解析機能	アプリケーションデータから XML の生成、XML からアプリケーションデータの生成を行う	劉宏	

	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	

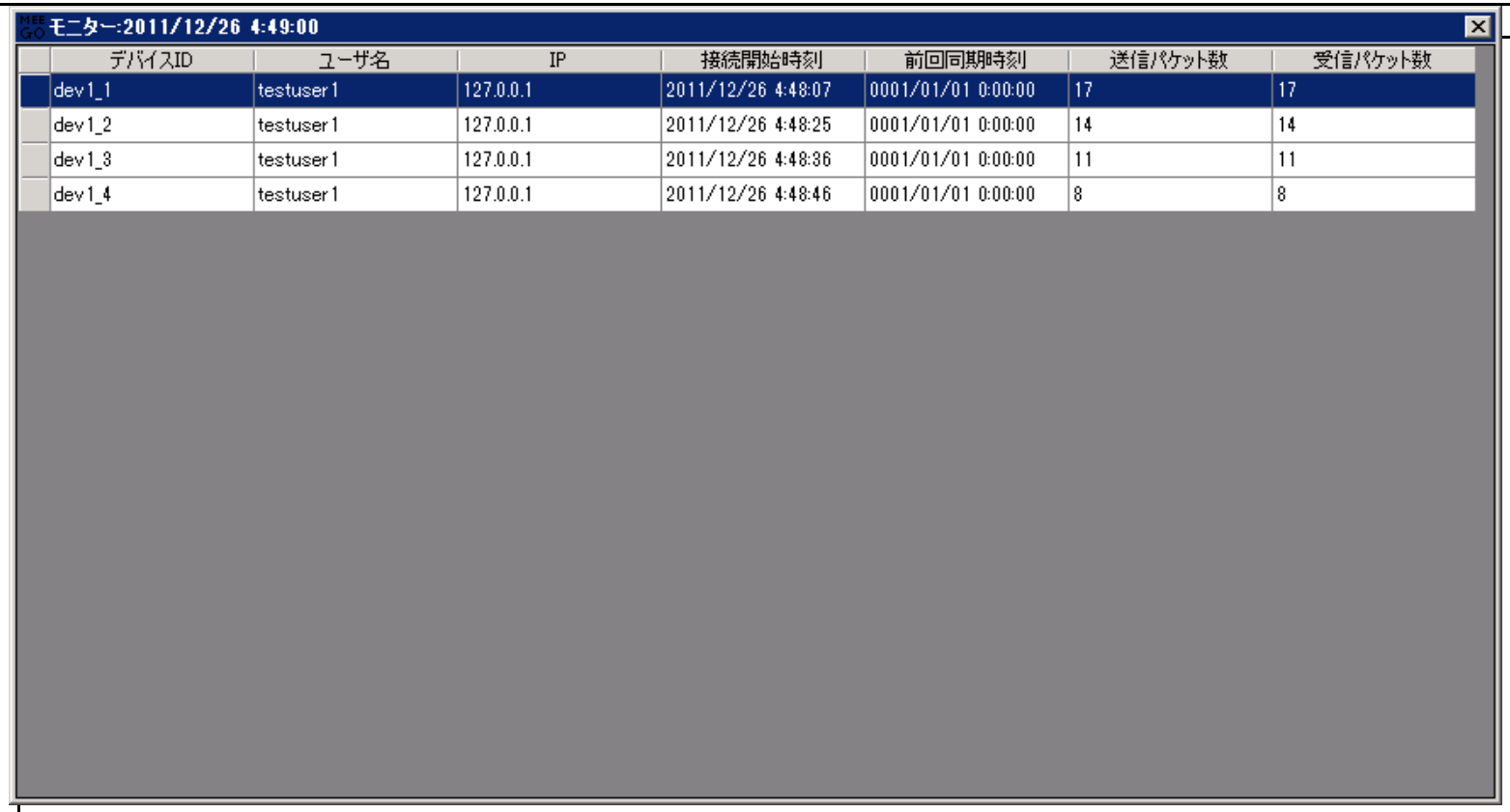
ID	namespace	クラス名	分類	名前	機能	戻り値	パラメータ
1		SystemDBOperator	接続関係	int Initiate(string connString)	データベース接続 open	int 0：ok 0以外：エラーコード	connString　型：string
2				void Dispose()	終了処理 DBConnection closeなど	なし	なし
3			ユーザ関係	int AddUser(string userName,string password,string mail,string deviceId)	新しいユーザを登録する ※DB作成も含む	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

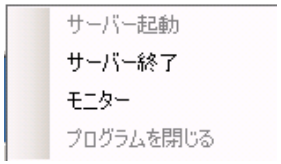
UserDBOperator

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					



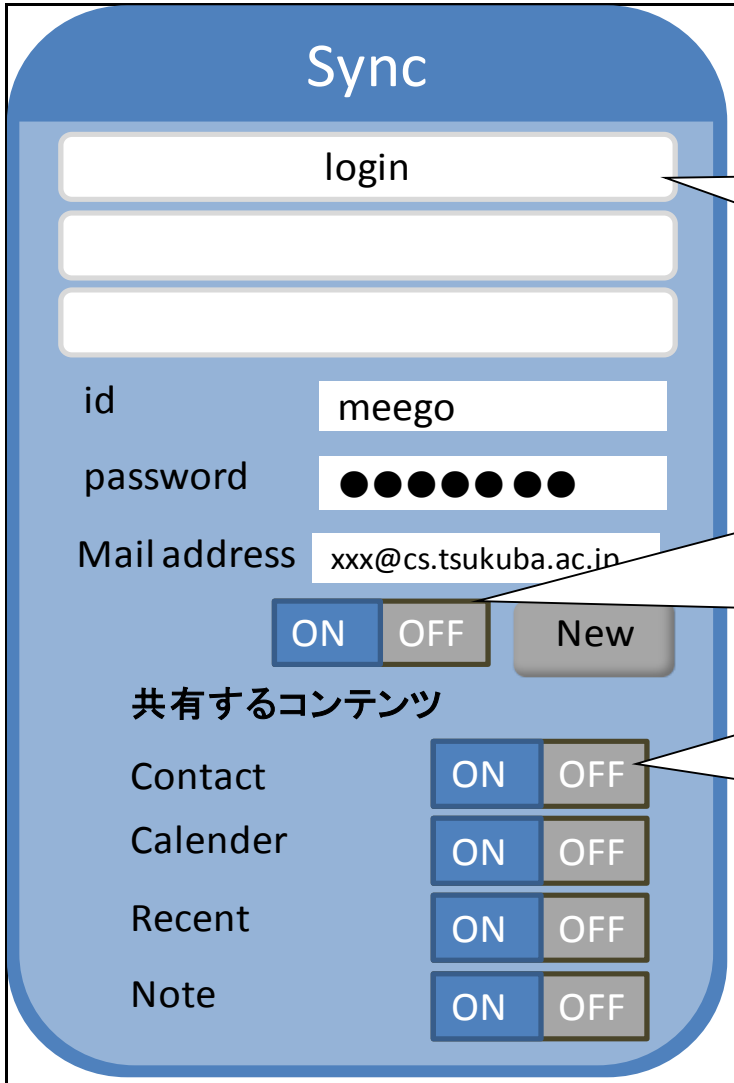
画面定義書	画面名	モニタリング画面	作成日	2011/12/15	作成者	劉宏超
	システム名	MeeGoデータ共有システム	更新日		更新者	
	備考					

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					



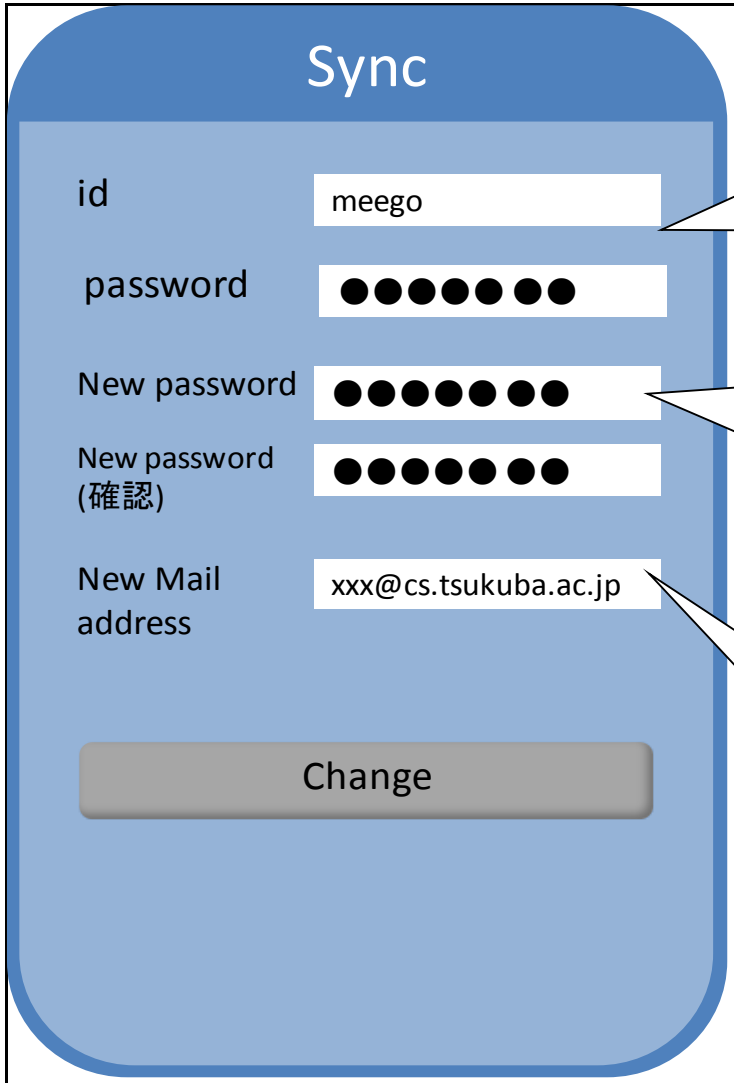
login、logout、login errorなどのステータスを直近三回分表示する

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	asswordス入力ボックス	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。

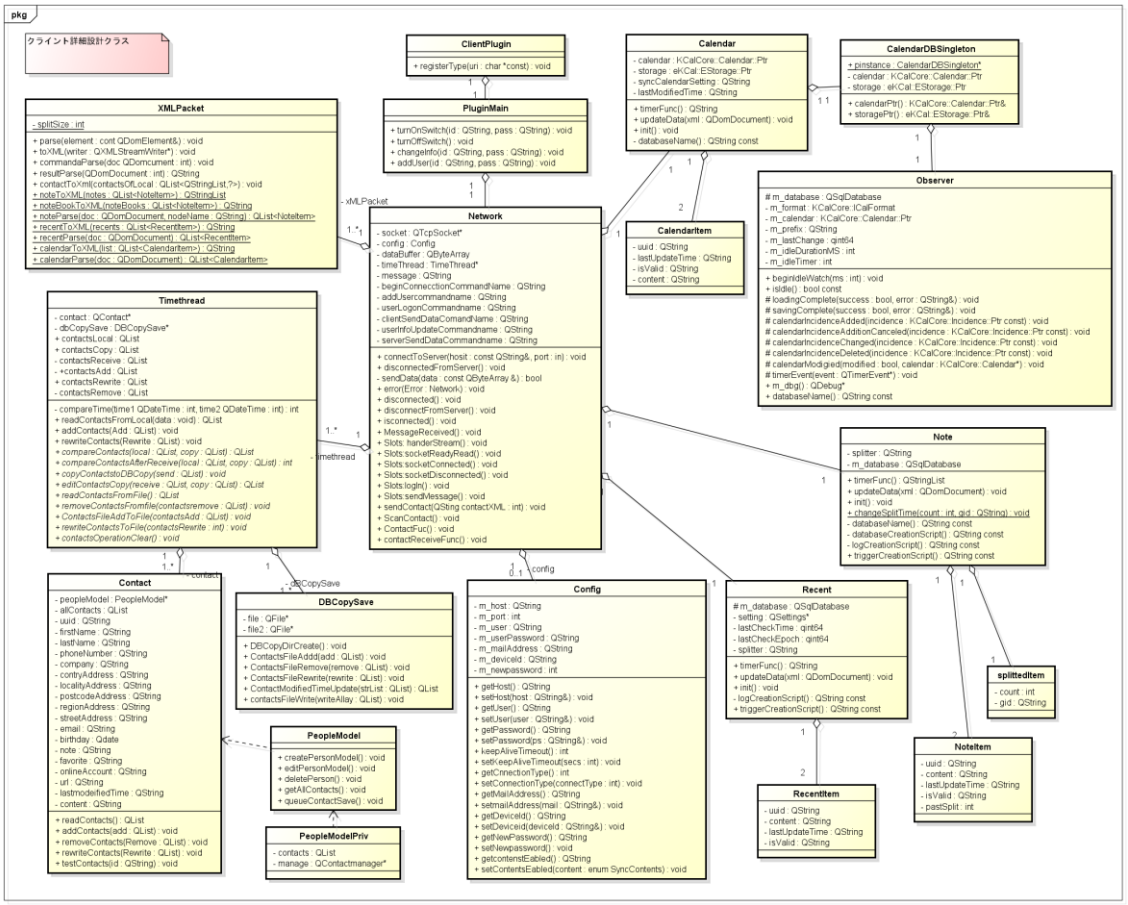


図 1 設計クラス図

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

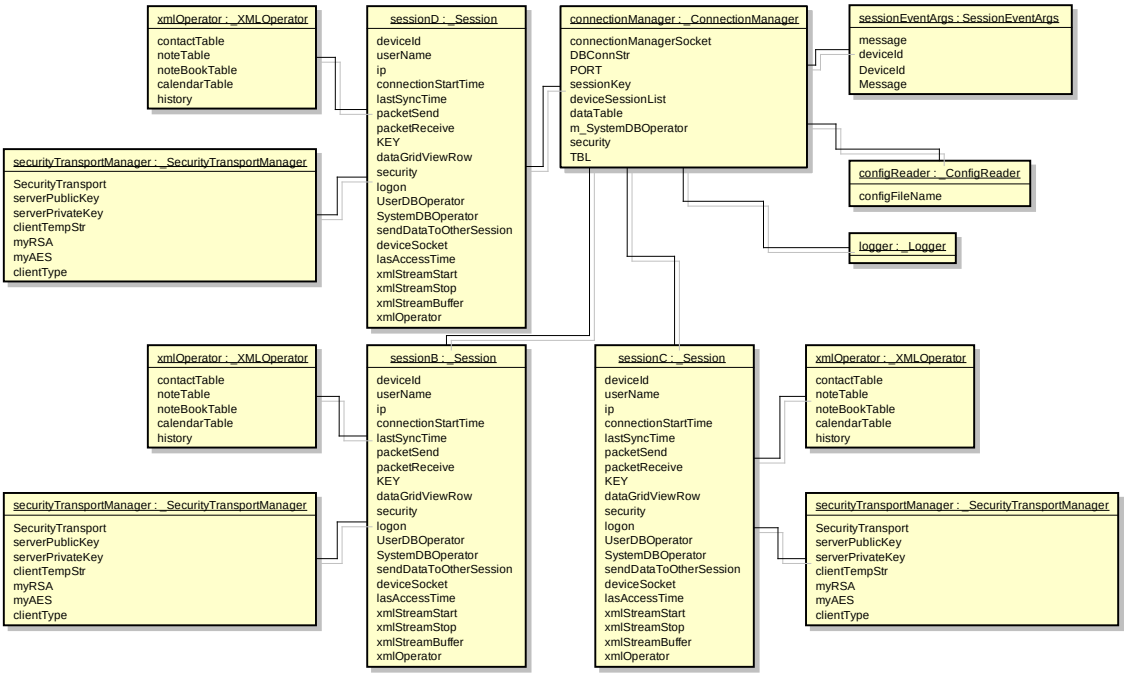
クラス詳細				
クラス名	**_SessionEventArgs**			
親クラス	EventArgs			
可視性	Public			
クラス概要				
他のでデバイスへ送信する用イベントクラスである				
インクルード				
1	System	7		
2	System.Collections.Generic	8		
3	System.Linq	9		
4	System.Text	10		
5		11		
6		12		
メソッド				
可視性	メソッド名	型	引数	説明、その他
public	SessionEventArgs	void	deviceId message	サーバ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

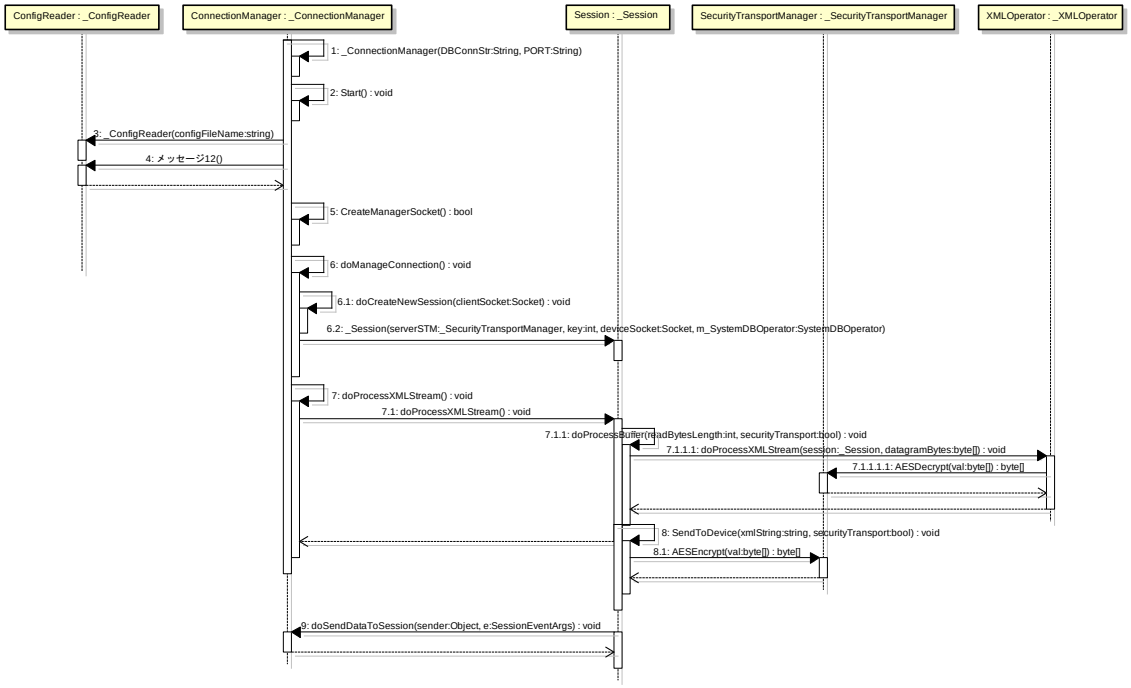
3　インスタンス図

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。

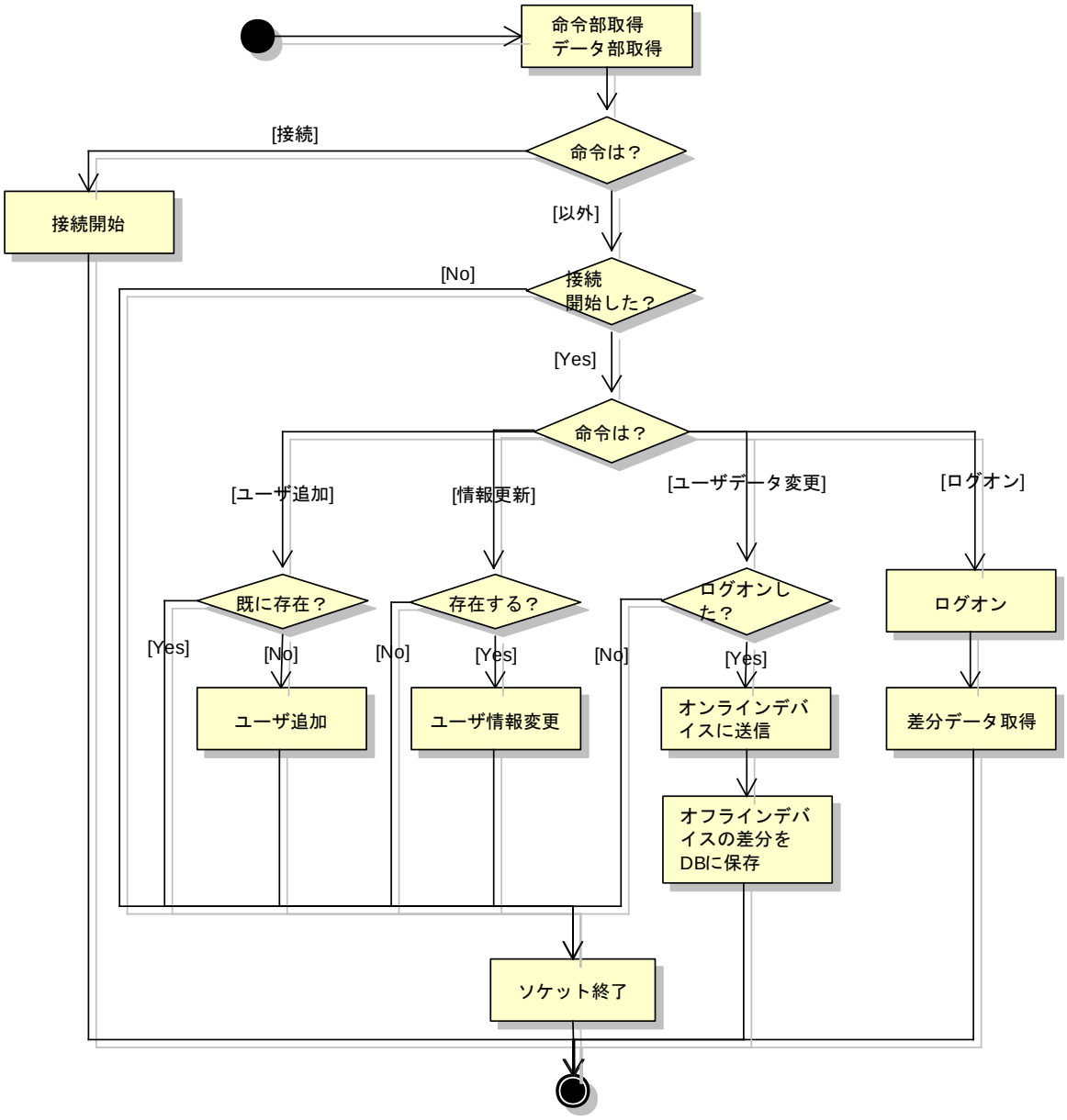
[DBConnStr]

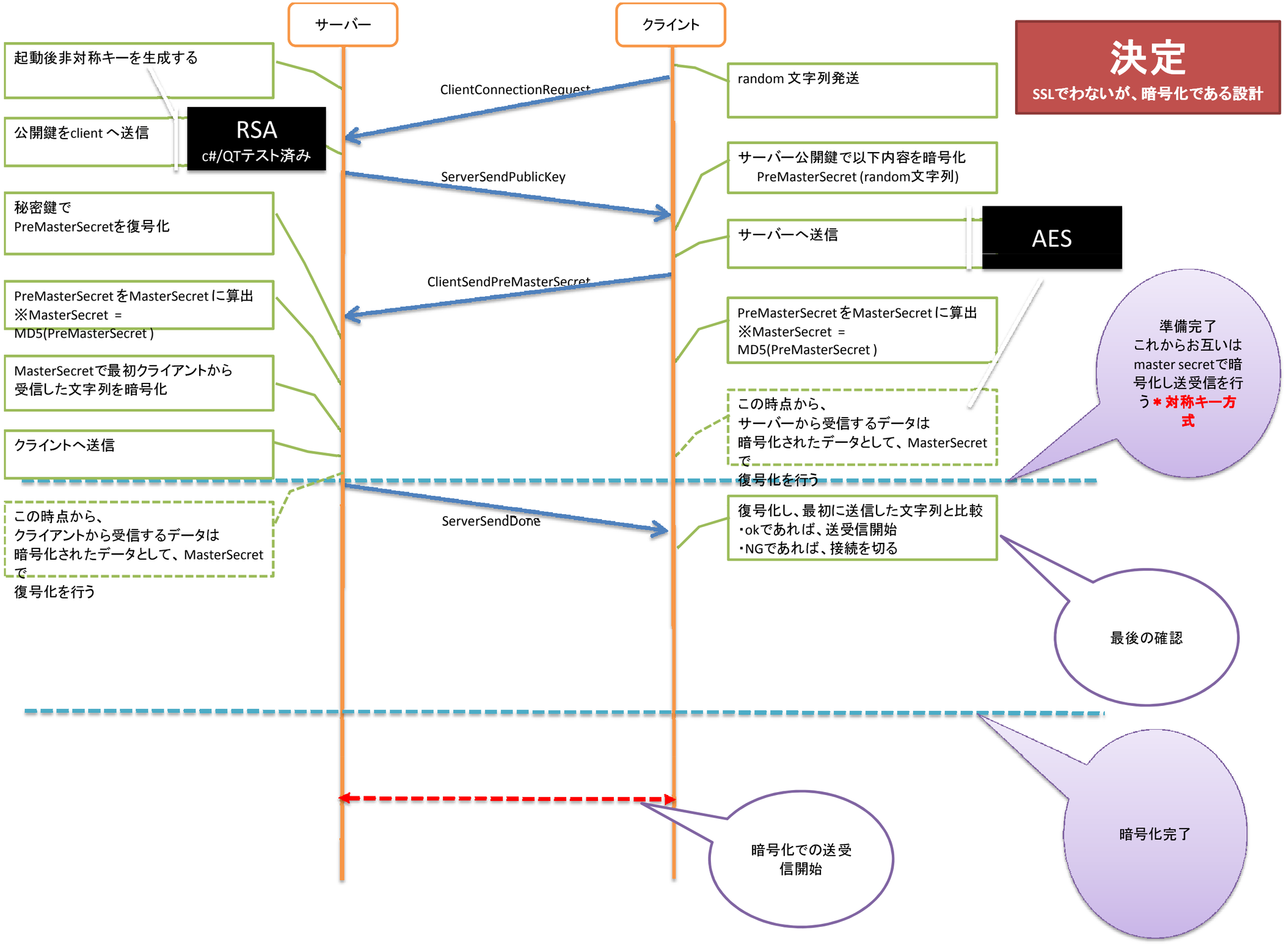
Server=[IP Address];UserId=[DBusername];Password=[DB pass];CharSet=utf8;Port=[port №];Max Pool Size=[xxx];

[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			

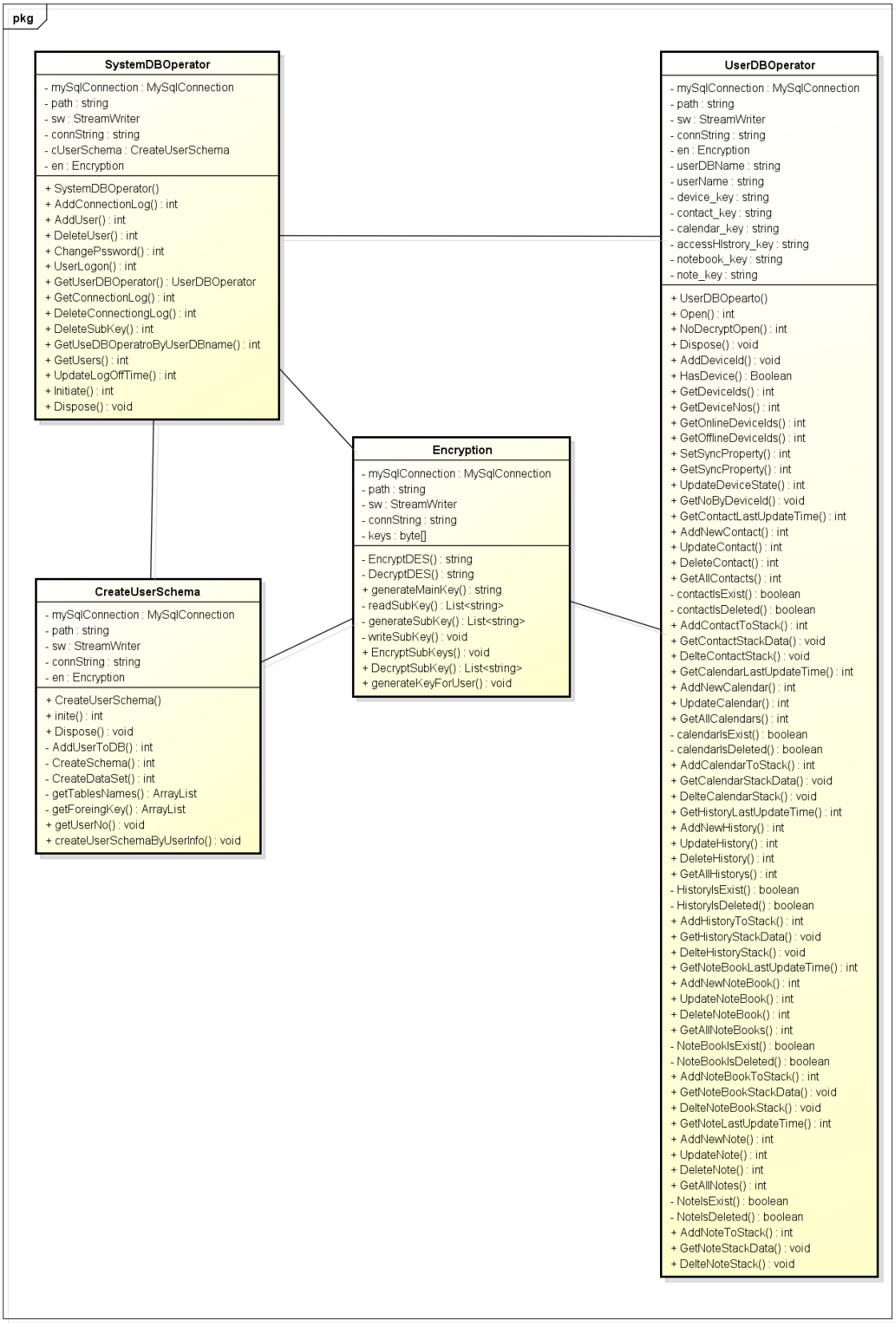
1.5　定義（用語、略語）

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

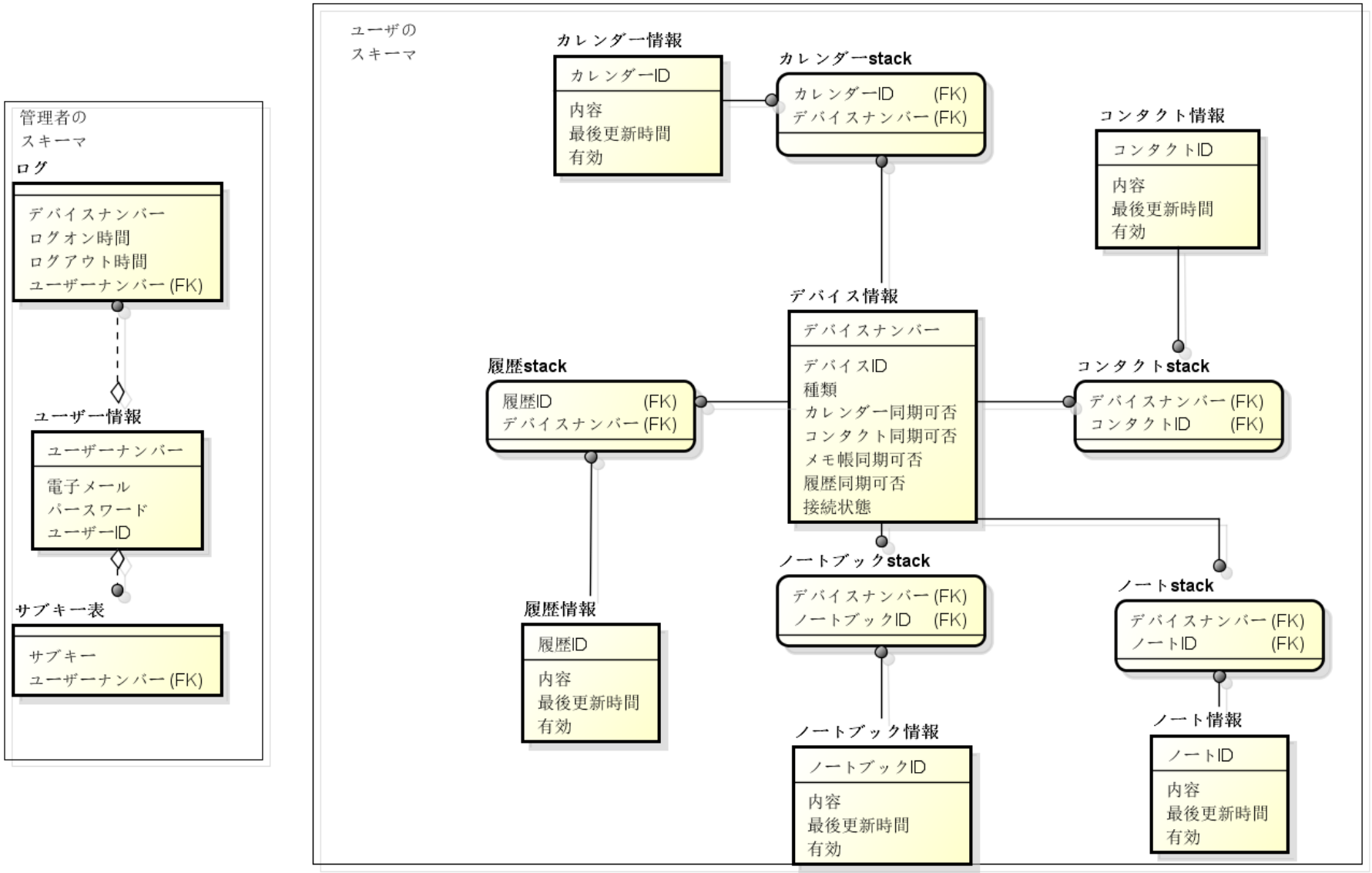
ID	標準用語	フリガナ	物理名	略語	標準用語説明
D001	ユーザー	ユーザー	USER		Meego データ共有システムサービスに登録したユーザー
D002	電子メール	デンシメール	E-MAIL		ユーザーの電子メールアドレス
D003	パースワード	パースワード	PASSWORD		
D004	デバイス	デバイス	DEVICE		ユーザーを持っている meego デバイス
D006	ログイン	ログイン	LOGIN		
D007	ログアウト	ログアウト	LOGOUT		
D008	ID	アイディ	ID		DB 上のテーブルで一意に管理するための値
D09	時間	ジカン	TIME		
D010	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	

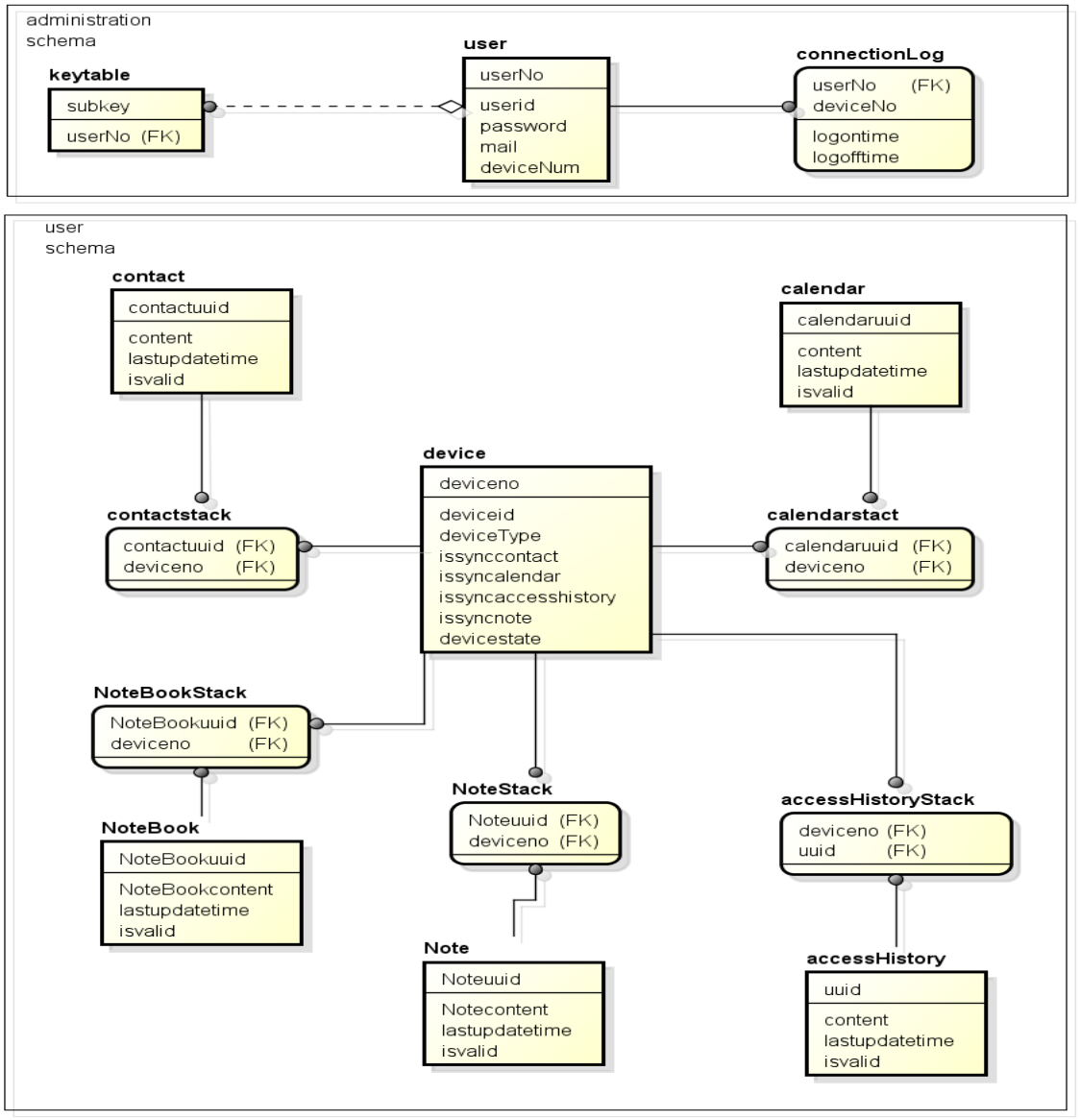
2.3　エンティティー覧

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																																