Cloud to Device Messaging Framework

http://code.google.com/intl/ja/android/c2dm/

Translated++ by 丸山不二夫
@maruyama097

Cloud to Device Messaging C2DMとは何か

- 開発者が、サーバからデータを、Androidデバイス上のアプリケーションに送るのを助けるサービス。
- アプリを更新したりユーザ・データを取得するためにサーバに直接コンタクトするように、モバイル・アプリに働きかけるシンプルで軽量なメカニズムをサーバに提供する。
- C2DMは、メッセージのキューイングとターゲットのデバイス上のターゲットのアプリに対するメッセージ配送のすべてを管理する。

はじめに

C2DMの特徴

- 開発者がサーバを立てれば、自分のAndroidアプリに軽量なメッセージを送ることが出来る
- ただ、このメッセージ・サービスは、メッセージを通じて大量のコンテンツを送るようには設計されていない。
- むしろ、アプリにサーバ上に新しいデータがあるので、それを取りに行けと告げるといった使い方をすべきである。

C2DMの特徴

- C2DMは、メッセージ配送の順序については、保証をしない。
- だから、もし、このサービスをインスタント・メッセージのアプリに使おうと思ったら、新しいメッセージが到着したことを知らせるために使うことはできるが、実際のメッセージを送るのには用いるべきではない。

C2DMの特徴

- Android上のアプリを、メッセージを受け取るために走らせておく必要はない。
- メッセージが届いた時、Androidは、Intent Broadcastのメカニズムを用いて、アプリを起動する。
- それを可能にするためには、アプリは、Broadcast Receiverとパーミッションの設定を、あらかじめ適切に行っていなければならない。

C2DMの特徴

- C2DMは、特別のユーザ・インターフェースや特別のメッセージ・データの処理方法を提供するわけではない。
- C2DMは、受け取ったままのデータを、そのままアプリケーションに渡すだけである。データの処理は、全面的にアプリケーションに委ねられている。
- メッセージを受け取って、アプリはNotificationをポストするかもしれないし、ユーザ・インターフェースを表示するかもしれないし、黙ってデータの同期をするかもしれない。

C2DMの特徴

- C2DMは、2.2以上のAndroidが走っているデバイスが必要である。かつ、Android Marketアプリがインストールされていなければならない。
- ただ、Marketを通じて配布されるアプリに、その利用が制限されているわけではない。

C2DMの特徴

- C2DMは、Googleのサービスとの接続が必要である。
- ユーザは、モバイル上で、Googleアカウントの設定をしておかなければならない。

C2DMアーキテクチャー概説

基本用語と基本コンセプト

- C2DMで用いられている基本的な用語とコンセプトは、次の二つのカテゴリーに分かれる。

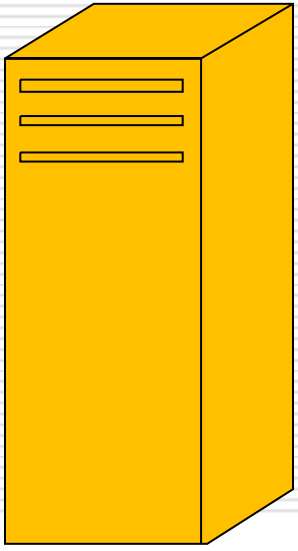
- **Component**：C2DMで働く物理的な実体。
- **Credentials**：C2DMのそれぞれ異なった段階で働く、IDとトークン。すべての当事者の認証と、メッセージが正しく配送されることを保証するもの。

Components

- Component
  - Mobile Device
  - Third-Party Application Server
  - C2DM Servers

Google
C2DMサーバ

C2DMを構成する
3つのコンポーネント



アプリケーション・サーバ

モバイル・デバイス

Mobile Device

- C2DMを利用するAndroidアプリが走るデバイス。
- このデバイスは、Androidマーケットをインストールした、Android2.2デバイスでなければならない。
- また、このデバイスは、少なくとも一つのログインしたGoogleアカウントを持たなければならない。

Third-Party Application Server

- 開発者が、自分のアプリにC2DMを実装する際に、その一部として設定したアプリケーション・サーバ。
- Third-Party Application Serverは、デバイス上のAndroidアプリに、C2DM Server経由で、データを送る。

C2DM Servers

- Third-Party Application Serverからメッセージを取り出し、それをデバイスに送る役割を果たすGoogleのサーバ。

Credentials

- Credentials
  - Sender ID
  - Application ID
  - Registration ID
  - Google User Account
  - Sender Auth Token

Sender ID

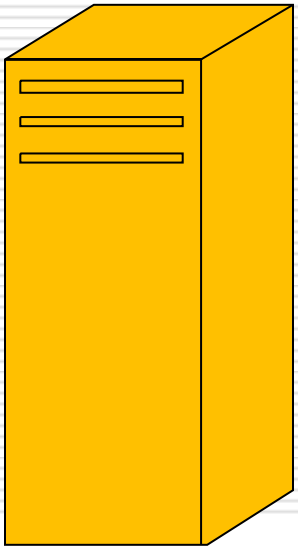
- アプリの開発者にひもづいたemailアカウント
- このSender IDは、デバイスにメッセージを送ることを許されたAndroidアプリの登録プロセスで利用される。
- このIDは、典型的には、個人アカウントというよりは、roleベースである。
- 例えば、 my-app@gmail.com のように。

Application ID

- メッセージを受け取ると登録されているアプリケーションは、manifest中のパッケージ名で同定される。
- これによって、メッセージが正しいアプリをターゲットとすることが保障される。

Google
C2DMサーバ

デバイス中のアプリは、IDを持つ



Sender ID

Application ID

アプリケーション・サーバ

モバイル・デバイス

Registration ID

- メッセージを受け取ることを許しているAndroidアプリに対して、C2DMサーバから、発行されるID。
- いったん、アプリがこのIDを取得すると、アプリは、それをthird-party application serverに送り、サーバは、それをメッセージを受け取ることを登録しているデバイスの同定に利用する。
- 換言すれば、Registration IDは、特定のデバイス上の特定のアプリに結びついている。


Google
C2DMサーバ
1. Register
2. Registrtion ID
3. Registrtion ID
Android
アプリ
アプリケーション・サーバ
モバイル・デバイス

Google User Account

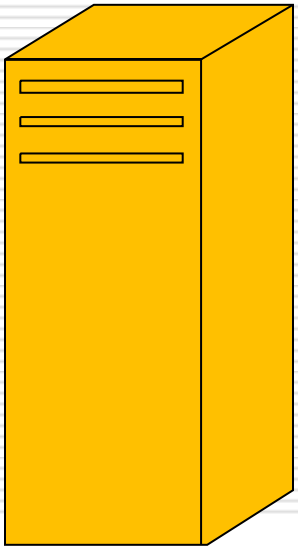
- C2DMが機能するためには、モバイル・デバイスは、少なくとも一つの、ログインしているGoogleアカウントを持つ必要がある。

Sender Auth Token

- Third-party application server 上に保存されたClientLogin Auth トークン。
- これによって、アプリケーション・サーバは、Googleのサービスにアクセスする権限を持つ。
- このトークンは、メッセージを送るPOSTリクエストのヘッダに含まれている。

Google
C2DMサーバ

POST
ClientLogin Auth
トークン



アプリケーション・サーバ

モバイル・デバイス

C2DMのライフサイクル

1. C2DMへのアプリの登録
2. メッセージの送信
3. メッセージの受信

C2DMの主要なプロセス

- **C2DMへの登録**: モバイル・デバイス上で走るAndroidアプリが、メッセージを受け取るという登録を行う。
- **メッセージを送る**:アプリケーション・サーバがメッセージをデバイスに送る。
- **メッセージを受け取る**: Androidアプリが、C2DMサーバからメッセージを受け取る。

C2DMへの登録（1）
REGISTER Intentでの受信登録

- アプリケーションが最初に、メッセージング・サービスを必要とした時には、アプリは、C2DMサーバに対して、Register Intent (com.google.android.c2dm.intent.REGISTER)を送りつける。
- このregister Intent は、Sender ID (アプリケーションにメッセージを送る権限を持つアカウントで、典型的にはアプリの開発者によって設定されたemailアドレス）とApplication IDを含んでいる。

Google
C2DMサーバ

1. Register

```
public static void register(Context context,
    String senderId) {
  Intent registrationIntent = new Intent(
    "com.google.android.c2dm.intent.REGISTER");
  registrationIntent.setPackage(GSF_PACKAGE);
  registrationIntent.putExtra(
    EXTRA_APPLICATION_PENDING_INTENT,
    PendingIntent.getBroadcast(context, 0,
        new Intent(), 0));
  registrationIntent.putExtra(EXTRA_SENDER,
    senderId);
  context.startService(registrationIntent);
}
```

モバイル・デバイス

C2DMへの登録(2) Registaration IDの取得

- 登録が成功すれば、C2DMサーバは、REGISTRATION Intentをブロードキャストする。それによってアプリケーションは、Registration IDを得る。
- アプリは、このIDを後で使うために保存しておく。Googleは、定期的に、Registration IDを更新する。だから、REGISTRATION Intentが何度も呼ばれることがあることを理解してアプリを設計しなくてはならない。アプリは、順次、それに対応できなくてはならない。

Google
C2DMサーバ

2. Registrtion ID

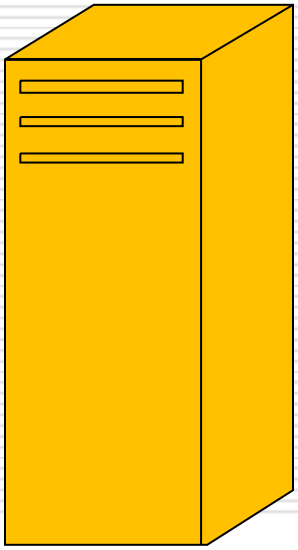
```
public final void onHandleIntent(Intent intent) {
  try {
    Context context = getApplicationContext();
    if (intent.getAction().equals(
     "com.google.android.c2dm.intent.REGISTRATION")
    handleRegistration(context, intent);
```

モバイル・デバイス

C2DMへの登録(3) Registaration IDの送付

- 登録を完了するために、アプリケーションはRegistration IDをアプリケーション・サーバに送る。アプリケーション・サーバは、典型的には、それをデータベースに保存しておく。
- Registration IDは、アプリケーションが明示的に自分を登録から外した時か、あるいは、GoogleがアプリのRegistration IDを更新する時まで、存続する。
- 提供されているサンプルには、アプリケーション・サーバ側のソースコードはない！

Google
C2DMサーバ



3. Registrtion ID

アプリケーション・サーバ

モバイル・デバイス

メッセージの送信の前提

- アプリケーション・サーバがメッセージを送るためには、次のような設定が必要である。

- アプリが、registaration ID を持っていて、特定のデバイスがメッセージを受け取ることが出来るようになっていること。
- アプリケーション・サーバが、そのregistration IDを保持していること。

メッセージの送信　ClientLogin token

- アプリケーション・サーバがメッセージを送るためには、もう一つ用意しておくべきことがある。
- ClientLogin authorization tokenである。これは、そのアプリの為に、開発者があらかじめアプリケーション・サーバにセットアップしておくべきもので、デバイスにメッセージを送る際に利用される。

メッセージの送信 ClientLogin token

- ClientLogin tokenは、アプリケーション・サーバに、特定のAndroidアプリにメッセージを送る権限を与える。
- アプリケーション・サーバは、一つの特定アプリについて一つのClientLogin tokenを持ち、複数のregistration IDを持つ。
- それぞれのregistration IDは、特定のアプリについて、メッセージング・サービスを利用する特定のデバイスを表現している。

Google
C2DMサーバ

メッセージの送信

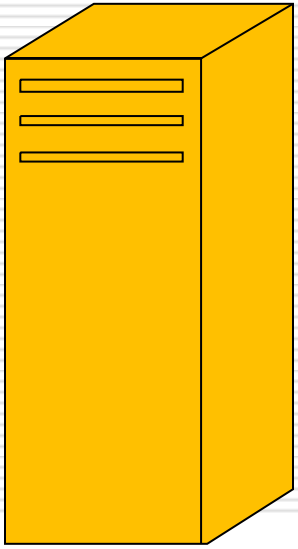
POST **https://android.apis.google.com/c2dm/send**

registration_id

collapse_key

data.<key>

ClientLogin Auth



アプリケーション・サーバ

モバイル・デバイス

メッセージ送信のシーケンス

1. アプリケーション・サーバは、GoogleのC2DMサーバにメッセージを送る。
2. Google側は、デバイスが立ち上がっていない場合には、メッセージをキューに入れて蓄えておく。
3. デバイスがオンラインになったら、Googleは、デバイスにメッセージを送る。

メッセージ送信のシーケンス

4. デバイス側では、システムは、特定のアプリに対して、適当なパーミッション設定をし、そのターゲットのアプリだけがメッセージを取得できるように、Intent Broadcastを使って、メッセージをブロードキャストする。これで、アプリは立ち上がる。メッセージを受け取るために事前に動いている必要はない。
5. アプリは、メッセージを処理する。処理が複雑なものであれば、wake lockを取得して、バックグラウンドのServiceで処理する。

メッセージ受信のシークエンス

1. システムは、送られたメッセージを受け取り、メッセージのpayloadから、生のkey/valueペアを直接に受け取る。
2. システムは、ターゲットのAndroidアプリに、key/valueペアを、com.google.android.c2dm.intent.RECEIVE IntentのExtrasに詰めて渡す。
3. Androidアプリは、RECEIVE Intentから、keyを用いて生のデータを取り出し、データを処理する。

Google
C2DMサーバ

Send Message

```
public final void onHandleIntent(Intent intent) {
  try {
    Context context = getApplicationContext();
    if (intent.getAction().equals(
     "com.google.android.c2dm.intent.REGISTRATION"
      handleRegistration(context, intent);
     } else if (intent.getAction().equals(
     "com.google.android.c2dm.intent.RECEIVE")) {
      onMessage(context, intent);
```

モバイル・デバイス

ユーザに見えること

- モバイル・デバイスのユーザがC2DMを含んだアプリをインストールすると、Android マーケットから、そのアプリはC2DMを利用しているがそれを許諾するかという情報が流れる。
- ユーザは、こうしたアプリをインストールする時C2DMを使うことを承認しなければならない。
- アプリの実装によっては、ユーザはメッセージの受け取りには登録をしないと選択も可能。
- アンストールは、登録解除と同じ効果を持つ。

C2DMを使ったAndroidアプリを書く

- C2DMを使ったAndroidアプリを書くためには「第三者のアプリケーション・サーバの役割」で述べるような仕事を実行できるアプリケーション・サーバを立てなければならない。
- ここでは、C2DMを使ったクライアント・アプリを作るためのステップを説明する。
- C2DM Frameworkには、特定のユーザ・インターフェースが定義されているわけではないことに注意せよ。どのようにメッセージを処理するかは、開発者の自由である。

クライアント・アプリ作成の二つのステップ

- アプリがC2DMを使うために必要なパーミッションを含んだManifestを作成する。
- Javaコードを実装する。C2DMを利用するためには、この実装は次のものを含んでいる必要がある。
  - registration serviceを開始/停止するコード
  - 次のIntentのReceiver
    - com.google.android.c2dm.intent.C2D_MESSAGE
    - com.google.android.c2dm.intent.REGISTRATION.

Manifestを作る

AndroidManifest.xmlファイル

- 全てのアプリは、そのルート・ディレクトリに、AndroidManifest.xmlという名前のファイルを持たなければならない。
- このファイルは、Androidシステムに対するアプリの重要な情報を含んでいる。Androidシステムは、アプリのコードを実行する以前に、必ずこの情報を持たなければならない。

C2DMアプリのAndroidManifest.xmlの情報

- アプリが、メッセージを登録、受信するのに必要なパーミッション。
  - com.google.android.c2dm.permission.RECEIVE
- アプリが、アプリケーション・サーバにreceiverキーを送るのに必要なパーミッション。
  - android.permission.INTERNET
- 他のアプリが、アプリのメッセージを登録したり受信したりしないようにする。
  - applicationPackage + ".permission.C2D_MESSAGE

C2DMアプリのAndroidManifest.xmlの情報

- 次の二つのIntentに対するReceiver。このIntentのCategoryセットは、applicationPackage
  - com.google.android.c2dm.intent.RECEIVE
  - com.google.android.c2dm.intent.REGISTRATION
- C2DMフレームワークだけがメッセージを送れるようにReceiverには、次のパーミッションが必要。
  - com.google.android.c2dm.SEND
- メッセージの登録・受信は、Intentとして実装されていることに留意すること。

C2DMアプリのAndroidManifest.xmlの情報

- もしも、C2DMの特徴が、アプリの機能にとって厳しいものであれば、マニフェスト中で次のように指定すること。
  - android:minSdkVersion="8"
- こうすることによって、C2DMがうまく走らないような環境では、アプリがインストールされないようにできる。

- 受信された C2D_MESSAGE Intentは、アプリケーション・サーバによって送られたKey/Valueペアを、Extrasとして持っている。
- この中で、特別なキーは、collapse_keyである。このキーは、メッセージの送り手によって指定され、オフラインのデバイスを待っている間に、メッセージの処理を可能にする。

C2DMのManifestのサンプル

```
<manifest package="com.example.myapp" ...>

<!– このアプリだけが、メッセージと登録結果を受け取る -->
<permission
android:name="com.example.myapp.permission.C2D_MESSAGE" android:protectionLevel="signature" />
<uses-permission
android:name="com.example.myapp.permission.C2D_MESSAGE" />

<!– このアプリは、登録とメッセージ受信のパーミッションを持つ -->
<uses-permission
android:name="com.google.android.c2dm.permission.RECEIVE" />
```

```
<!-- registration idをサーバに送る -->
<uses-permission
android:name="android.permission.INTERNET" />

<!-- C2DMサーバだけが、このアプリにメッセージを送れる。もしもこの
パーミッションがセットされていないと、他のアプリも、メッセージを作れてし
まうことになる。 -->
<receiver android:name=".C2DMReceiver"
android:permission="com.google.android.c2dm.permission.SEND">
    <!– 実際のメッセージを受け取る -->
    <intent-filter>
        <action android:name=
            "com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.example.myapp" />
    </intent-filter>
```

```
    <!-- registration id を受け取る  -->
    <intent-filter>
        <action android:name=
         "com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.example.myapp" />
    </intent-filter>
</receiver>
```

C2DMに登録する

Androidアプリは、メッセージを受け取る前にC2DMサーバに登録する必要がある。

com.google.android.c2dm.intent.REGISTER

登録用のIntent

- 登録の為に、Intent (com.google.android.c2dm.intent.REGISTER)を、次の2つのパラメータ付きで送る。
- **sender** は、このアプリにメッセージを送る権限を持ったアカウントのIDである。典型的には、アプリの開発者によって設定されたアカウントのemailアドレスである。
- **app** は、アプリのIDで、PendingIntentとともにセットされ、登録サービスが、アプリの情報を取り出すことを可能とする。

登録用のコードのサンプル

```
Intent registrationIntent = new Intent(
        "com.google.android.c2dm.intent.REGISTER");
registrationIntent.putExtra("app",
        PendingIntent.getBroadcast(this, 0, new Intent(), 0);
        // boilerplate
registrationIntent.putExtra("sender", emailOfSender);
startService(registrationIntent);
```

登録の作業は、アプリが、アプリケーション・サーバにregistration IDを送るまでは、完了しない。
アプリケーション・サーバは、registration IDを、特定のデバイス上で走るターゲットのアプリにメッセージを送るために利用する。

登録解除のコード

```
Intent unregIntent = new Intent(
        "com.google.android.c2dm.intent.UNREGISTER");
unregIntent.putExtra("app",
        PendingIntent.getBroadcast(this, 0, new Intent(), 0));
startService(unregIntent);
```

登録結果の処理

com.google.android.c2dm.intent.REGISTRATION

REGISTRATION Intent

- このREGISTRATION Intentの主な使い方は、アプリに、registration IDを受け取ることを可能にすることである。
- このIntentは、いつでも送ることが出来る。Googleは、定期的に receiver IDを更新することがある。
- このIntentを、registration IDとともに受け取ったアプリは、アプリケーション・サーバがregistration IDを受け取ったことを保証しなければならない。

REGISTRATION Intent

- そうした保証は、registration ID を保存し、それをサーバに送ることによって行われる。
- もしも、ネットワークがダウンしていたり、エラーが起きた時には、アプリは、ネットワークが回復した時、あるいは、アプリが次にスタートした時に、registration ID の再送を試みるべきである。
- アプリは、登録の現在の状態を把握して、もしもそれが完全に終わっていなければ、再度、登録を試みなければならない。

REGISTRATION Intent

- REGISTRATION Intentは、登録が完了しなかった場合には、エラー・パラメータを生成する。こうした場合、アプリは、後で、再試行すべきである。（1秒後、2秒後、4秒後、8秒後、16秒後…といった、exponential back offで）
- アプリの登録が解除された時、登録解除のExtrasパラメータを持った、REGISTRATION Intent が送信される。

REGISTRATION Intent エラー

- **SERVICE_NOT_AVAILABLE**
  デバイスが、レスポンスを読むことが出来ないか、サーバから500/503が返った時。
  アプリは、exponential back offのスタイルで再試行しなければならない。
- **ACCOUNT_MISSING**
  デバイス上に、Google accountがない。
  アプリは、アカウント・マネージャを開いて、Googleアカウントを追加するようにユーザに要求する。デバイス側で修正。

REGISTRATION Intent エラー

- **AUTHENTICATION_FAILED**
  パスワードが違っている。
  アプリは、ユーザにパスワードの再入力を求める。デバイス側で修正。
- **TOO_MANY_REGISTRATIONS**
  ユーザが、あまりに多くのアプリを登録している。アプリは、ユーザに他のいくつかのアプリのア人ストールを要求する。デバイス側で修正。

REGISTRATION Intent エラー

- **INVALID_SENDER**
  Senderのアカウントが認識できない。
- **PHONE_REGISTRATION_ERROR**
  Googleへの電話の登録が正しくない。この電話は、現在、C2DMをサポートしていない。

- アプリケーション・サーバがメッセージを送った時にはいつでも、アプリは、REGISTRATION Intentのbroadcastを受け取る。
- ただ、registration IDが存在しなかったり、正しくなかったりすることがある。
  - アプリが最初に走った時には、まだregistration IDs を持っていない。
  - アプリが登録解除されていれば、registration IDs はない。
  - C2DM サーバは、定期的にregistration IDs を更新している。

コード・サンプル

```
public void onReceive(Context context, Intent intent) {
    if (intent.getAction().equals(
        "com.google.android.c2dm.intent.REGISTRATION")) {
        handleRegistration(context, intent);
    } else if (intent.getAction().equals(
        "com.google.android.c2dm.intent.RECEIVE")) {
        handleMessage(context, intent);
     }
 }
```

コード・サンプル

```
private void handleRegistration(Context context, Intent intent) {
    String registration = intent.getStringExtra("registration_id");
    if (intent.getStringExtra("error") != null) {
        // 登録に失敗したら、後で再試行する
    } else if (intent.getStringExtra("unregistered") != null) {
        // 登録が解除されたら、新しいメッセージは拒否する
    } else if (registration != null) {
       // アプリケーション・サーバにregistration IDを送る
       // これは、別スレッドで行われるべき
       //これが終わってはじめて登録作業は終了である。
    }
}
```

受信したデータの処理

com.google.android.c2dm.intent.RECEIVE

- C2DM サーバが、アプリケーション・サーバからメッセージを受け取った時、C2DMは、メッセージから生のkey/value ペアを抜き出して、それをAndroidアプリに渡す。
- その時、**com.google.android.c2dm.intent.RECEIVE** Intent が用いられる。
- アプリは、Extrasから、キーでデータを取り出して、それを処理する。

```
protected void onReceive(Context context, Intent intent) {
    String accountName = intent.getExtras().
        getString(Config.C2DM_ACCOUNT_EXTRA);
    String message = intent.getExtras().
        getString(Config.C2DM_MESSAGE_EXTRA);
    if (Config.C2DM_MESSAGE_SYNC.equals(message)) {
        if (accountName != null) {
            if (Log.isLoggable(TAG, Log.DEBUG)) {
                Log.d(TAG, "Messaging request received for account "
                                + accountName);
            }
            ContentResolver.requestSync(
                new Account(accountName,
                    SyncAdapter.GOOGLE_ACCOUNT_TYPE),
                JumpNoteContract.AUTHORITY, new Bundle()
             );
        }
    }
}
```

アプリの開発とテスト

- C2DMアプリを開発・テストするためには、必要なGoogleのサービスを含んだAndroid2.2システム・イメージ上で、実行・デバッグを行う必要がある。
- 実機上で開発・デバッグを行うためには、Marketアプリを含んだAndroid2.2システム・イメージが走っているデバイスが必要である。
- Androidエミュレータ上で開発・テストを行うためには、Android SDKとAVD Managerを使って、SDKにGoogle APIs Add-OnのAndroid2.2バージョンをダウンロードする必要がある。特に、“Google APIs by Google Inc, Android API 8”という名前のコンポーネントをダウンロードする必要がある。その後、システム・イメージを使って、AVDをセットアップする。
- もしも、C2DMの特徴が、アプリの機能にとって厳しいものであれば、マニフェスト中でandroid:minSdkVersion=“8”と指定する。こうすることによって、C2DMがうまく走らないような環境では、アプリがインストールされないようにできる。

第三者のアプリケーション・サーバの役割

アプリケーション・サーバの機能

- C2DMの機能を持つクライアント・アプリを書く前に、次のような基準を満たすHTTPSサーバが必要である。
  - クライアントとコミュニケーションできること。
  - C2DMサーバにHTTPリクエストを送れること
  - リクエストを処理し、必要に応じてキューのデータを処理できること。例えば、exponential back offが可能である事。
  - ClientLogin Auth tokenとクライアントのregistration IDを保存できること。

アプリケーション・サーバからのメッセージの送信

- アプリケーション・サーバが、アプリにメッセージを送ることが出来るためには、その前に、アプリから、registration ID を受け取っていなくてはならない。
- メッセージを送るとき、アプリケーション・サーバは **https://android.apis.google.com/c2dm/send** に、POSTリクエストを送る。

アプリケーション・サーバからのPOSTに含まれるフィールド

- **registration_id**
  端末上のAndroidアプリから取得したregistration ID。必須。
- **collapse_key**
  デバイスがオフラインの時など、最後のメッセージだけがクライアントに送られるように、類似したメッセージのグループを折りたたむために使う、任意の文字列。端末がオンラインに復帰した時、沢山のメッセージが送られないためのもの。必須。メッセージの順序の保証はないので、本当に「最後」のメッセージではないかもしれないことに注意

アプリケーション・サーバからのPOSTに含まれるフィールド

- **data.<key>**
  key/valueペアで表現されたPayloadデータ
  もし、あれば、<key>を持ったアプリケーション・データとして、Intentの中に含まれることになる。オプショナル。
  key/valueペアの数に制限はないが、メッセージの全体のサイズには、制限がある。

アプリケーション・サーバからのPOSTに含まれるフィールド

- **delay_while_idle**
  もし含まれていれば、デバイスが動いていないとき、メッセージをすぐには送らないことを示す。オプショナル。
  サーバは、デバイスがアクティブになるのを待って、それぞれのcollapse_key毎に、その値の、最後のメッセージを送信する。
- **Authorization**:
  GoogleLogin auth=[AUTH_TOKEN]
  ClientLogin Authトークン。必須。
  クッキーは、ac2dm サービスに関連する。

アプリケーション・サーバへのレスポンス 200

- id=[ID of sent message]
- Error=[error code]
  - **QuotaExceeded** — あまりに多くのメッセージを送っている。少し後に再試行せよ。
  - **DeviceQuotaExceeded** — 特定のデバイスに、あまりに多くのメッセージを送っている。少し後に再試行せよ。
  - **InvalidRegistration** — registration_idが存在しないか、間違っている。このデバイスへのメッセージ送信を中止すべき。

アプリケーション・サーバへのレスポンス 200 （続き）

- **NotRegistered —** registration_idは、もはや有効ではない。例えば、ユーザがアプリをアンストールしたか、notificationを切った場合とか。このデバイスへのメッセージ送信を中止せよ。
- **MessageTooBig —** メッセージのペイロードが大きすぎる。制限を確認して、メッセージのサイズを小さくせよ。
- **MissingCollapseKey —** Collapse keyは、必須である。リクエストにCollapse keyを含めよ

アプリケーション・サーバへのレスポンス　503/401

- **503**　–　サーバは、一時的に利用できない。後で、レスポンス中のRetry-Afterヘッダを使って、再試行する。
  アプリケーション・サーバは、exponential back offを実装すべき。問題をおこしたサーバは、ブラックリストに載せられる危険もある。
- **401**　–　ClientLogin AUTH_TOKEN　が、正しくない。