BaaS@rakuza

SDK リファレンスマニュアル

Android 版

Ver2.2.0

第 3 版

PEOPLE SOFTWARE

内容

クイックスタート

BaaS@kuraza SDK for Android の利用方法

このページでは、BaaS@rakuza SDK for Android をお客様の環境で利用するための設定を行います。

手順書作成環境

当手順書は以下の環境で作成しています。
お客様の環境のバージョンによっては設定方法が異なる可能性があります。

- JDK1.7
- Android Studio 2.1
- OS X Yosemite

Android Studio

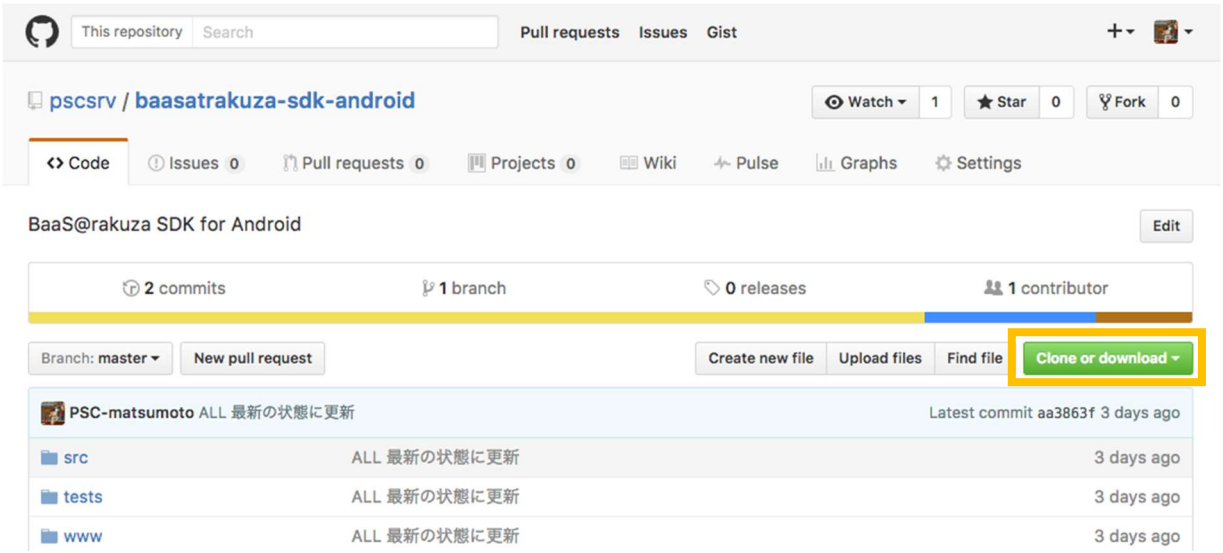
ダウンロードページからダウンロードし、任意のディレクトリに展開します。

Android Developer Tool (ADT)

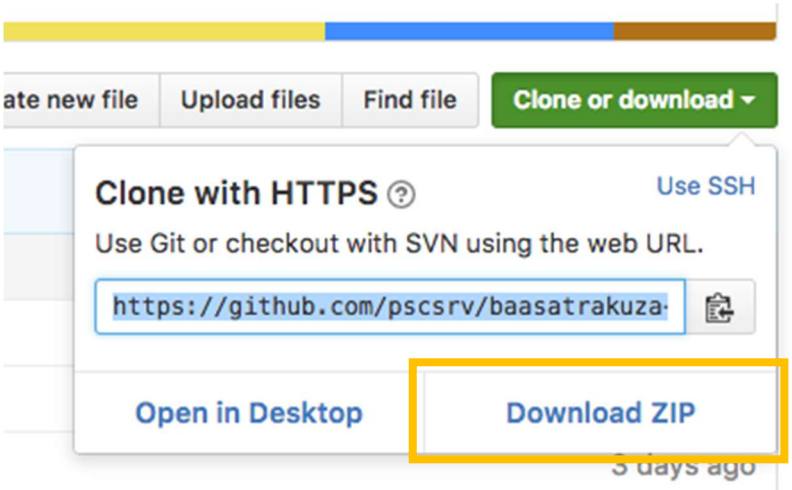
Android Studio 起動後、Android Studio のウィザードに従い、必要な ADT をインストールします。

BaaS@rakuza SDK for Android のダウンロード

最新の SDK は GitHub にて配布しています。
https://github.com/pscsrv/baasatrakuza-sdk-android へアクセスして「Clone or download」をクリックします。



クリック後に表示されるポップアップウィンドウの「Download ZIP」をクリックします。



お使いの PC に ZIP 形式で SDK がダウンロードされます。

ダウンロードした SDK の zip ファイルを、お使いの PC 上の任意のディレクトリに展開します。

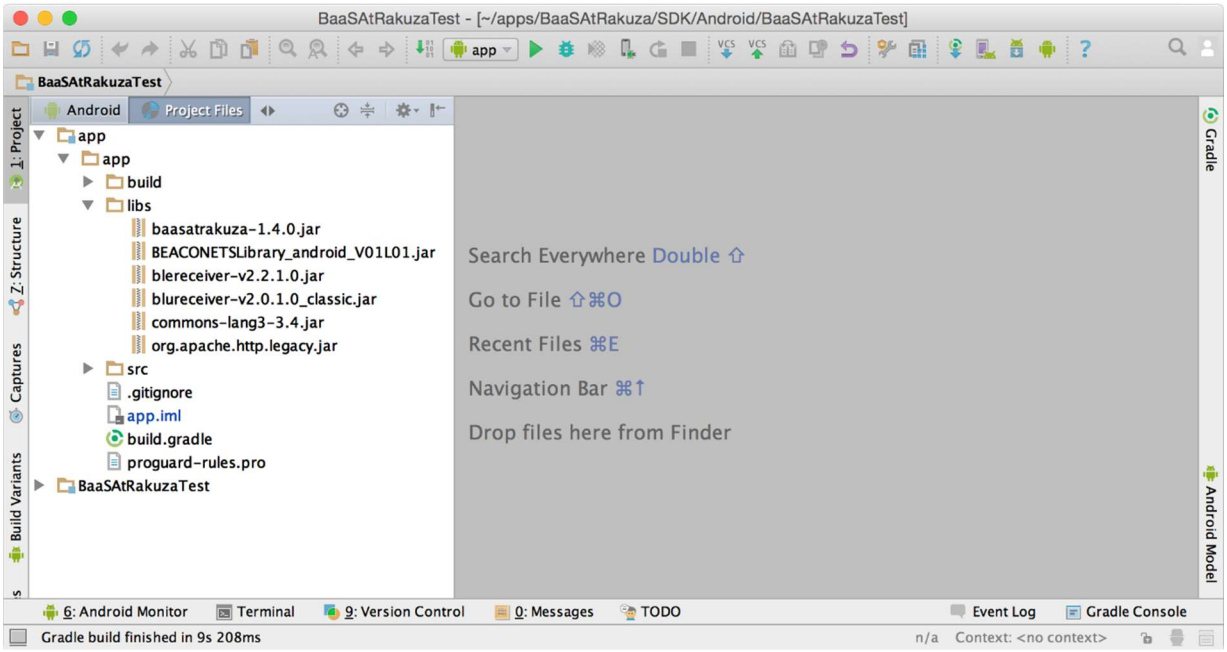
提供ファイルの構成は以下になります。

baasatrakuza-sdk-android
- docs
  - javadoc.zip
  - BaaSAtRakuzaSDK リファレンスマニュアル_Android_x.pdf
- libs
  - BaaSAtRakuzaSDK_for_Android.jar

プロジェクトへの設定

BaaS@rakuza SDK をプロジェクトに追加する

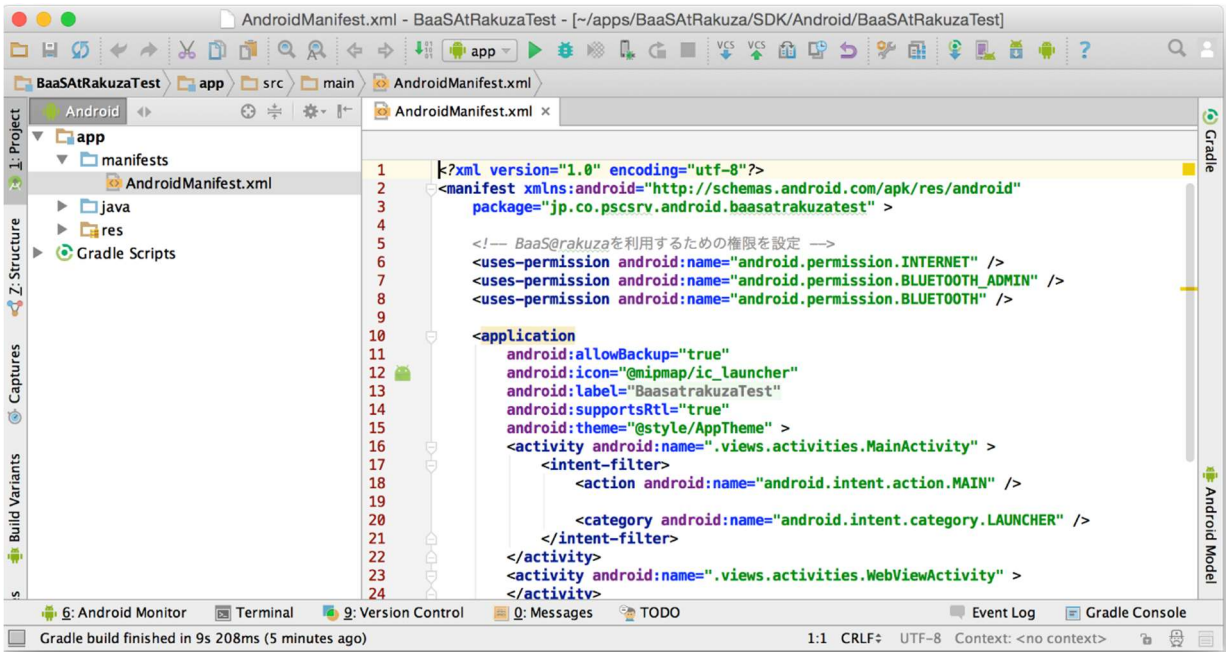
画面左にあるパッケージ・エクスプローラー内の
BaaSAtRakuza を使用したいプロジェクトの libs フォルダ内にダウンロードした jar ファイルをコピーします。



AndroidManifest.xml に権限の追加をする

作成したプロジェクト内にある AndroidManifest.xml に以下のパーミッションを追加します。
（追加場所は<application>タグの直前に記述してください。）

- <uses-permission android:name="android.permission.INTERNET"/>
- <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
- <uses-permission android:name="android.permission.BLUETOOTH" />
- <uses-permission android:name="android.permission.INTERNET" />



以上で BaaS@rakuza を利用する環境が整いました。

API を利用するための初期化処理

BaaS@rakuza SDK for Android を利用する際には、 RKZClient クラスのシングルトンインスタンスを利用します。
以下の処理をアプリ起動時におこなうことで、BaaS@rakuza の API を利用することが出来るようになります。

RKZClient クラスの初期化

ここでは、BaaS@rakuza SDK for Android を使用するうえ重要な RKZClient クラスの初期化を説明します。
BaaS@rakuza SDK for Android では RKZClient クラスの初期化は最初に呼び出される Activity で初期化する事を推奨していますが、どの場所で初期化を行っても構いません。
作成したプロジェクト内にある以下のソースを変更していきます。

- src / 自分で定義したパッケージ / MainActivity.java

MainActivity を開き、onCreate メソッド内の setContentView(R.layout.activity_main);の下に以下のコードを追加します。

```
RKZClient  client  =  RKZClient.getInstance();
client.initialize ( getApplicationContext() , "配布したテナントキー", new OnInitializeListener( ) {
    @Override
    public  void  onInitialize ( boolean isSuccess , RKZResponseStatus  rkzStatus ) {
        //  初期化に成功した場合
        if ( isSuccess ) {
            // client 変数を用いて SDK の機能を使用できます。
            Toast.makeText ( getApplicationContext( ) ,
                    "RKZClient クラスの初期化に成功しました。", Toast.LENGTH_SHORT ) . show( );
        }
     }
}
```

また、ソースコード上部に import 文として下記の記述を追加してください。(すでに同じ記述がある場合は追加しないでください。)

```
package jp.co.pscsrv.android.baasatrakuzasample;

import jp.co.pscsrv.android.baasatrakuza.client.RKZClient;
import jp.co.pscsrv.android.baasatrakuza.core.RKZResponseStatus;
import jp.co.pscsrv.android.baasatrakuza.listener.OnInitializeListener;
import jp.co.pscsrv.android.baasatrakuzasample.R;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;


public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        RKZClient client = RKZClient.getInstance();
        client.initialize(getApplicationContext(), "XXXXXXXX", new OnInitializeListener() {

            @Override
            public void onInitialize(boolean isSuccess, RKZResponseStatus rkzStatus) {
                // 初期化に成功した場合
                if(isSuccess) {
                    // client変数を用いてSDKの機能を使用できます。
                    Toast.makeText(getApplicationContext(),
                            "RKZClientクラスの初期化に成功しました。", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

上記プログラムを実行後、端末画面下に"RKZClient クラスの初期化に成功しました。"と数秒間だけ表示されれば OK です。

もし、表示されない場合は再度配布したテナントキーの確認を行ってください。
また Android Studio 上でエラーが出ている場合は以下の確認を行ってください。

- libs フォルダ内に指定の jar ファイルを追加する。
- import 文の記述が間違っている（記述されていない）
- onCreate メソッド内に初期化用の記述が間違っている（記述されていない）

データ管理

データ管理機能を利用する

データ管理機能は、BaaS@rakuza 標準オブジェクト以外の情報を管理する基本的な仕組みを提供します。
このページでは、データ管理機能を利用する実装例を紹介します。

複数レコード取得する（キー未指定）

複数レコード取得の場合、検索条件とソート条件を指定することができます。指定可能な条件については

データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について

を参照してください。

複数レコード取得（キー未指定）は RKZClient の getDataList で行います。

お気に入り情報登録（addFavoriteToObjectData）したデータを取得する場合は、拡張属性を指定します。

取得に成功した場合は、取得したレコードのコードをソート順にアラートで表示します。
取得に失敗した場合はエラー内容をアラートで表示します。

```
// 検索条件データを作成する
// オブジェクト内の code の値が"9999"か"0005"か"0032"か"9493"のデータを取得する。
List<RKZApiSearchCondition> searchCondition = new ArrayList<RKZApiSearchCondition>( );
List<String> targetBeacon = new ArrayList<String> ( );

// 取得したい値を追加していく
targetBeacon.add( "9999" );
targetBeacon.add( "0005" );
targetBeacon.add( "0032" );
targetBeacon.add( "9493" );

// 複数指定の物を探し出す命令を指す RKZApiSearchCondition.IN を指定し、
// 検索を掛ける項目名を第２引数で指定する。
// 最後に数値を指定する。
searchCondition.add( new RKZApiSearchCondition( RKZApiSearchCondition.IN, "code",  targets );
```

```
// ソート条件データを作成する
// オブジェクト内の名前を降順でソートする場合
List<RKZApiSortCondition> sortCondition = new ArrayList<RKZApiSortCondition>( );

// RKZApiSortCondition クラスのコンストラクタで
// 第１引数に降順を示す RKZApiSortCondition.DESC を指定し、
// 第２引数にソートを掛ける項目名の name を指定する。
sortCondition.add ( new RKZApiSortCondition ( RKZApiSortCondition.DESC, "name" ) );
// 検索条件、ソート条件を指定しないで取得する
// 取得したいオブジェクト ID は必須項目です
String objectId = "object1";

// 検索条件、ソート条件を使用する場合は、
// 検索条件を第２引数に、　ソート条件を第３引数に指定してください。
RKZClient.getInstance().getDataList( objectId, searchCondition, sortCondition, new OnGetRKZObjectDataListListen
er( ) {
    @Override
public void onGetRKZObjectDataList(List<RKZObjectData> dataList, RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            for ( RKZObjectData rKZObjectData : dataList ) {
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getCode( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getName( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getShortName( ) );
                Log.d( "OnGetRKZObjectDataListener", Integer.toString(rKZObjectData.getSortNo( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getAttributesValue("code");
            }
        } else {
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getMessage( ) );
        }
    }
}
```

検索条件について

BaaS@rakuza SDK では複数レコード取得時に検索条件を指定する事ができます。　※一部指定できないものもあります。
検索条件に指定可能なタイプは以下になります。
また、複数検索条件を指定した場合は、各検索条件を AND 条件で指定します。

定数名	条件
RKZSearchCondition	検索値のいずれかに該当する
RKZSearchConditionNotIn	検索値のいずれにも該当しない
RKZSearchConditionEqual	検索値と一致
RKZSeachConditionNotEqual	検索値と一致一致しない
RKZSearchConditionLikeBefore	検索値に前方一致する
RKZSearchConditionLikeAfter	検索値に後方一致する
RKZSearchConditionLikeBoth	検索値に部分一致する
RKZSearchConditionBetweenInclude	検索した検索値の範囲内（検索値含む）
RKZSearchConditionBetweenExclude	指定した検索値の範囲内（検索値を含まない）
RKZSearchConditionLessThanInclude	指定した検索値以上
RKZSearchConditionGreaterThanInclude	検索した検索値以下
RKZSearchConditionLikeOr	楽座項目「チェックボックス」専用
RKZSearchConditionWithFavorite MyFavoriteOnly	お気に入り登録された情報のみ ※お気に入り情報取得時のみ指定
RKZSearchConditionWithFavorite NotMyFavorite	お気に入り登録されていない情報のみ ※お気に入り情報取得時のみ指定
RKZSearchConditionWithFavoriteAll	お気に入り登録有無に関わらず全て ※お気に入り情報取得時のみ指定
RKZSearchConditionReadedNewsAlreadyRead	既読のお知らせ情報のみ ※お知らせ既読未読情報取得時のみ指定
RKZSearchConditionReadedNewsNoRead	未読のお知らせ情報のみ ※お知らせ既読未読情報取得時のみ指定
RKZSearchConditionReadedNewsAll	お知らせ既読未読に関わらず全て ※お知らせ既読未読情報取得時のみ指定

ソート条件について

BaaS@rakuza SDK では複数レコード取得時にソート条件を指定することもできます。　※一部指定できないものもあります。
ソート条件に設定可能なタイプは以下になります。
また、複数ソート順を指定した場合は、追加順でソート順を決定します。

定数名	条件
RKZSortCondition.ASC	昇順
RKZSortCondition.DESC	降順

お気に入り登録を行ったオブジェクトを並び替えるための専用メソッドは以下になります。

定数名	条件
RKZSortCondition. initWithSortTypeForFavoriteUpdateDate	お気に入り登録された日付でソートを設定します。
RKZSortCondition. initWithSortTypeForFavoriteCount	お気に入り登録された件数でソートを設定します。

メソッドを呼び出す際に指定するパラメータ引数は、ASC、DESC のどちらかを指定します。
お気に入りの並び替え条件が指定できるメソッドは、

- getDataList
- getPaginateDataList

の 2 メソッドになります。

1 レコード取得する（キー指定）

レコード取得（キー指定）は RKZClient の getData で行います。データは RKZObjectData として返却されます。

取得に成功した場合はコールバックメソッドの第 1 引数に検索結果のデータが渡されます。
取得失敗の場合は第 2 引数の isSuccess メソッドが false を返します。

```
//  キーを指定してレコードを取得するには
//  オブジェクト ID とデータのコード[code]が必要になります。
String  objectId = "object1";
String code         = "0001";

RKZClient.getData( objectId, code, new OnGetRKZObjectDataListener( ) {
@Override
    public void onGetRKZObjectData( RKZObjectData objectData, RKZResponseStatus rkzSatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            // 成功時
            Log.d( "OnGetRKZObjectDataListener", objectData.getCode( ) );
```

```
            Log.d( "OnGetRKZObjectDataListener", objectData.getName( ) );
            Log.d( "OnGetRKZObjectDataListener", objectData.getShortName( ) );
            Log.d( "OnGetRKZObjectDataListener", Integer.toString( objectData.getSortNo( ) ) );
            Log.d( "OnGetRKZObjectDataListener", objectData.getAttributesValue("code");
        } else {
            Log.d( "OnGetRKZObjectDataListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListener", rkzStatus.getMessage( ) );
        }
    }
}
```

オブジェクトデータを登録する

データオブジェクトへのデータ登録は RKZClient の addData で行います。
登録に成功したか失敗したかを取得することができます。

登録に成功した場合はコールバックメソッドの第１引数に"1001"が渡されます。
登録に失敗した場合はコールバックメソッドの第２引数の isSuccess メソッドが false を返します。

```
// RKZObjectData を作成し、登録に最低限必要なデータを設定します
RKZObjectData rKZData = new RKZObjectData( );

rKZData.setName( “名称” );
rKZData.setObjectId( “object001” );

RKZClient.addData( rKZData, new OnAddRKZObjectDataListener( ) {
    @Override
    public void onAddRKZObjectData( String status, RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            //  登録に成功した際の処理を記述
            Log.d( "OnAddRKZObjectDataListener", "登録成功" );
        } else {
            Log.d( "OnAddRKZObjectDataListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnAddRKZObjectDataListener", rkzStatus.getMessage( ) );
        }
    }
}
```

オブジェクトデータを編集する

データオブジェクトのデータ編集は RKZClient の editData で行います。
編集に成功したか失敗したかを取得することができます。

編集に成功した場合はコールバックメソッドの第１引数に文字列"1001"が渡されます。
編集に失敗した場合はコールバックメソッドの第２引数の isSuccess メソッドが false を返します。

```
// RKZObjectData を作成し、編集に必要なデータを設定します
RKZObjectData rKZData = new RKZObjectData( );

rKZData.setName("名称" );
rKZData.setObjectId("object1" );
rKZData.setCode("0001" );

RKZClient.editData( rKZData, new OnEditRKZObjectDataListaner( ) {
    @Override
    public void onEditRKZObjectData( String status,  RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            //  データ編集に成功した際の処理を記述
            Log.d( "OnEditRKZObjectDataListener", "編集成功" );
        } else {
            Log.d( "OnEditRKZObjectDataListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnEditRKZObjectDataListener", rkzStatus.getMessage( ) );
        }
    }
}
```

オブジェクトデータを削除する

オブジェクトのデータ削除は RKZClient の deleteData で行います。
編集に成功したか失敗したかを取得することができます。

削除に成功した場合はコールバックメソッドの第１引数に文字列"1001"が渡されます。
削除に失敗した場合はコールバックメソッドの第２引数の isSuccess メソッドが false を返します。

複数レコード削除の場合、検索条件を指定することができます。指定可能な条件については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
を参照してください。

```
String objectId = "object01";    // オブジェクト ID を指定します

// 第２引数には RKZApiSearchCondition を指定する。
RKZClient.getInstance().deleteData(objectId, null, new OnDeleteDataListener() {
  @Override
  public void onDeleteData(final Integer deleteCount, final RKZResponseStatus rkzResponseStatus) {
    // 復帰値を設定する
    if ( rkzStatus.isSuccess( ) ) {
        //  データ編集に成功した際の処理を記述
        Log.d( "OnDeleteDataDataListener", "編集成功" );
    } else {
        Log.d( "OnDeleteDataListener", rkzStatus.getStatusCode( ) );
        Log.d( "OnDeleteDataListener", rkzStatus.getMessage( ) );
    }
  }
});
```

階層付きレコードを複数取得する（キー未指定）

階層付きレコード取得（キー未指定）は
RKClient の getDataListWithRelationObjects で行います。データは RKZObjectData として返却されます。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について

を参照してください。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。
取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
String objectId = "object01";        // 検索対象のオブジェクト ID を指定します
Integer treeCount = 2;               // 2 階層分取得します

RKZClient.getDataListWithRelationObjects( objectId, treeCount, null, null, new OnGetRKZObjectDataListListener
( ) {
    @Override
    public void onGetRKZObjectDataList(List<RKZObjectData> dataList, RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            for ( RKZObjectData rKZObjectData : dataList ) {
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getCode( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getName( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getShortName( ) );
                Log.d( "OnGetRKZObjectDataListener", Integer.toString(rKZObjectData.getSortNo( ) );
                Log.d( "OnGetRKZObjectDataListener", rKZObjectData.getAttributesValue("code");
            }
        } else {
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getMessage( ) );
        }
    }
}
```

ページング機能を利用してレコードを取得する

データオブジェクトからレコードを取得するときに、ページング機能を利用すると
- 取得するレコードの開始位置
- 取得するレコードの件数

といったレコードを分割して取得するための条件を指定することができます。

取得結果には条件に該当したレコードのほかに、
- 指定された条件に該当したデータの総件数

も取得することができます。

ページングを利用してレコードを取得する場合は、
RKZClient の getPaginateDataList で行います。取得結果は RKZPagingData として復帰されます。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

お気に入り情報登録（addFavoriteToObjectData）したデータを取得する場合は、拡張属性を指定します。
拡張属性は null を指定しています。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。
取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
String objectId = "object01";        // 検索対象のオブジェクト ID を指定します
Integer limit = 10;                  // 取得するデータの件数を指定します
Integer offset = 10;                 // データの取得位置を指定します

RKZClient.getInstance().getPaginateDataList( objectId, limit, offset, null, null, null, new OnGetPagingDataLis
tener( ) {
    @Override
    public void onGetPagingData(final PagingData pagingData, final RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            for ( RKZObjectData rKZObjectData : dataList ) {
                Log.d( " OnGetPagingDataListener ", pagingData.getLimit() );
                Log.d( " OnGetPagingDataListener ", pagingData.getOffset() );
                Log.d( " OnGetPagingDataListener ", pagingData.getResultCnt() );
                Log.d( " OnGetPagingDataListener ", pagingData.getData() );
            }
        } else {
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getMessage( ) );
        }
    }
}
```

位置情報を利用してレコードを取得する

位置情報を利用してデータオブジェクトを取得することができます。

※注意点　位置情報を利用してデータを抽出する場合、取得対象となるデータオブジェクトに spot オブジェクトが関連付けされている必要があります。
spot オブジェクトが関連付けされているフィールドに対して検索を実行します。

位置情報を利用して取得するには、
RKZClient の getDataWithLocation（1 レコード取得）か、
getDataListWithLocation（複数レコード取得)で行います。

※注意点 spotFieldName は未指定でも取得可能です。データオブジェクトに複数の spot オブジェクトを関連付けている場合、検索する対象のフィールドを特定する場合に spotFieldName を指定します。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。
取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
String objectId = "object01";                    // 検索対象のオブジェクト ID を指定します
String code = "0001";                            // 取得するデータの件数を指定します
RKZLocation location = new RKZLocation();        // 位置情報を指定するためのオブジェクトを作成します
location.setLatitude(34.600917);                 // 緯度を指定
location.setLongitude(133.765784);               // 経度の指定

RKZClient.getDataWithLocation( objectId, code, location, null, new OnGetRKZObjectDataListener( ) {
    @Override
    public void onGetRKZObjectData(RKZObjectData objectData, RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            // 成功時
            Log.d( "OnGetRKZObjectDataListener", objectData.getCode( ) );
            Log.d( "OnGetRKZObjectDataListener", objectData.getName( ) );
            Log.d( "OnGetRKZObjectDataListener", objectData.getShortName( ) );
            Log.d( "OnGetRKZObjectDataListener", Integer.toString( objectData.getSortNo( ) ) );
            Log.d( "OnGetRKZObjectDataListener", objectData.getAttributesValue("code");
        } else {
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getMessage( ) );
        }
    }
}
```

データオブジェクトのフィールド定義を取得する

フィールド定義情報を取得するには、
RKZClient の getFieldDataList で行います。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。
取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
String objectId = "object01";       // オブジェクト ID を指定します
Boolean visibleOnly = true;         // 表示項目のみを取得するようにします

RKZClient.getFieldDataList( objectId, visibleOnly, new OnGetRKZFieldDataListListener() {
    @Override
    public void onGetRKZFieldDataList(List<RKZFieldData> datas, RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            // 成功時
            Log.d( " OnGetRKZFieldDataListListener ", datas.getFieldName( ) );
            Log.d( " OnGetRKZFieldDataListListener ", datas.getLabelStr( ) );
        } else {
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetRKZObjectDataListListener", rkzStatus.getMessage( ) );
        }
    }
}
```

お気に入り登録されたオブジェクトを取得する

登録されたオブジェクトデータに対してお気に入り登録を行った場合、登録されたお気に入り情報を条件に指定して抽出することができます。

お気に入りの登録については
お気に入り管理 ＞ お気に入り情報を登録する
を参照してください。

お気に入り登録されたオブジェクトを検索する場合は、

- getDataList
- getPaginateDataList

のメソッドを利用します。お気に入り登録情報を含めてデータを取得する場合は、extensionAttributes 引数を指定します。お気に入り情報を抽出する際に指定できる extensionAttribute は以下のとおりです。

引数名	型	内容
userAccessToken	String	登録したお気に入り情報を抽出する場合必須。
showFavorite	Boolean	取得結果にお気に入り情報を付けて取得する場合に指定します。
showFavoriteSummary	Boolean	取得結果にお気に入りの総件数を付けて取得する場合に指定します。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。

取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
String objectId = "object1";
List<RKZSearchCondition> searchConditions = new ArrayList<>();
List<RKZSortCondition> sortConditions = new ArrayList<>();

// extensionAttribute 引数を設定
ObjectDataExtensionAttribute extensionAttribute = new ObjectDataExtensionAttribute();
extensionAttribute
    .setUserAccessToken(USER_ACCESS_TOKEN)
    .setShowFavorite(true)    // お気に入り情報を復帰する
    .setShowFavoriteSummary(true);  // お気に入り件数を復帰する

RKZClient.getInstance().getDataList(objectId,
        searchConditions,
        sortConditions,
        extensionAttribute,
        new OnGetRKZObjectDataListListener() {
    @SuppressLint("LongLogTag")
    @Override
    public void onGetRKZObjectDataList(List<RKZObjectData> rKZObjectDataList, RKZResponseStatus rKZResponseSta
tus) {
        if (rKZResponseStatus.isSuccess()) {
            for (RKZObjectData rKZObjectData : rKZObjectDataList) {
                Log.d( "OnGetRKZObjectDataListener", (String) ((Map) rKZObjectData.getAttributesValue("sys_fav
orite")).get("is_favorite") );
                Log.d( "OnGetRKZObjectDataListener", (String) ((Map) rKZObjectData.getAttributesValue("sys_fav
orite")).get("favorite_date") );
            }
        } else {
            Log.d( "OnGetRKZObjectDataListener", rKZResponseStatus.getStatusCode() );
            Log.d( "OnGetRKZObjectDataListener", rKZResponseStatus.getMessage() );
        }
    }
});
```

ユーザー管理

ユーザー管理機能を利用する

ユーザー管理機能は、アプリケーションでユーザーの情報を管理する基本的な仕組みを提供します。
このページでは、ユーザー管理機能を利用する実装例を紹介します。

ユーザー情報を登録する

ユーザー情報登録は RKZClient の registUser で行います。

登録に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
登録に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

ユーザー登録に成功した場合、userData に"user_access_token"が格納されて返却されます。"user_access_token"はユーザーに関連する情報を取得・変更する際にユーザーを特定するキーとして必ず必要となりますので、ユーザーを扱うアプリケーションを開発する場合は、"user_access_token"をアプリケーションの永続データ領域に保存しておくように実装して下さい。

```
//  登録するユーザー情報を作成します
User  user  = new User( );
//  登録したい情報を設定していきます
//  名前の設定
user.setFirstName( "ピープル" );
user.setLastName( "太郎" );

//  ユーザー登録 API の実行（RKZClient 変数は初期化済みとする）
RKZClient.registUser( user, new OnGetUserListener( ) {
    @Override
    public void onGetUser( User user,  RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            //  ユーザー登録成功
            //  成功時には引数の user 内に登録内容が格納されて返却されます
            Log.d( "OnGetUserListener", user.getFirstName( ) );
            Log.d( "OnGetUserListener", user.getLastName( ) );
            Log.d( "OnGetUserListener", user.getUserAccessToken( ) );
        } else {
            //  ユーザー登録失敗
            //  失敗時には rkzStatus にエラー情報が格納されて返却されます
```

```
            Log.d( "OnGetUserListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetUserListener", rkzStatus.getMessage( ) );
        }
    }
}
```

ユーザー情報を取得する

ユーザー情報取得は RKZClient の getUser で行います。

ユーザー情報取得に成功した場合はコールバックリスナの第１引数にユーザーデータが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// ユーザアクセストークンが必要
String  userAccessToken  = "userAccessTokenXXXX";

RKZClient.getUser( userAccessToken, new OnGetUserListener( ) {
    @Override
    public void onGetUser( User user, RKZResponseStaus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            // ユーザー情報取得成功時
            Log.d( "OnGetUserListener", user.getUserFirstName( ) );
            Log.d( "OnGetUserListener", user.getUserLastName( ) );
            Log.d( "OnGetUserListener", user.getMailAddress1( ) );
            Log.d( "OnGetUserListener", user.getTelNo1( ) );
        } else {
            // ユーザー情報取得失敗時
            Log.d( "OnGetUserListener", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetUserListener", rkzStatus.getMessage( ) );
        }
    }
}
```

ユーザー情報を編集する

ユーザー編集をメソッドから行う場合は、RKZClient の editUser で行います。

ユーザー編集に成功した場合はコールバックリスナの第１引数にユーザー情報を渡します。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// ユーザー編集にて、必須項目は編集するユーザー情報です。
RKZClient.editUser(user, new OnEditUserListener() {
  @Override
  public void onEditUser(User user, RKZResponseStatus rKZResponseStatus) {
    if (rKZResponseStatus.isSuccess()) {
      Log.d("OnEditUserListener", "編集成功");
    } else {
      Log.d("OnEditUserListener",rKZResponseStatus.getStatusCode());
      Log.d("OnEditUserListener",rKZResponseStatus.getMessage());
    }
  }
});
```

機種変更認証コードを発行する（必須項目のみ指定）

機種変更認証コード発行(必須項目のみ指定)は RKZClient の registModelChangeCode で行います。

機種変更認証コード発行に成功した場合は取得した認証コードと有効期限が復帰されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// 機種変更を行うための認証コードを生成します。
// ユーザーアクセストークンが必須です。
String userAccessToken = “userAccessTokenXXXX”;

RKZClient.registModelChangeCode(userAccessToken, new OnRegistModelChangeCodeListener(){
  @Override
  public void onRegistModelChangeCode(final String modelChangeCode,
                                      final Calendar limitDate,
                                      final RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess() {
      Log.d(“OnRegistModelChangeCodeListener”,modelChangeCode);        // 生成された認証コード
```

```
      Log.d(“OnRegistModelChangeCodeListener”,limitDate);        // 生成された認証コードの有効期限
    } else {
      // 生成失敗
      Log.d("OnEditUserListener",rKZResponseStatus.getStatusCode());
      Log.d("OnEditUserListener",rKZResponseStatus.getMessage());
    }
  }
});
```

機種変更コードを発行する（必須項目＋任意項目指定）

機種変更認証コード発行(必須項目+任意項目指定)は RKZClient の registModelChangeCode で行います。

機種変更認証コード発行に成功した場合は取得した認証コードと有効期限が復帰されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// 機種変更を行うための認証コードを生成します。
// ユーザーアクセストークンが必須です。
String userAccessToken = “userAccessTokenXXXX”;

// 任意項目(パスワード、桁数、有効期限)の設定を行います。
String password = “ユーザーが指定したパスワード”;  // 認証コード＋αのセキュリティ対策にパスワードを設定
Integer limitCount = 8;                          // 発行される認証コードの桁数
Integer limit = 60 * 24;                         // 認証コードの有効期限(単位：分)

RKZClient.registModelChangeCode(userAccessToken,
                                password,
                                limitCount,
                                limit,
                                new OnRegistModelChangeCodeListener(){
  @Override
  public void onRegistModelChangeCode(final String modelChangeCode,
                                      final Calendar limitDate,
                                      final RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess() {
      Log.d(“OnRegistModelChangeCodeListener”,modelChangeCode);  // 生成された認証コード
      Log.d(“OnRegistModelChangeCodeListener”,limitDate);        // 生成された認証コードの有効期限
    } else {
      // 生成失敗
      Log.d("OnEditUserListener",rKZResponseStatus.getStatusCode());
      Log.d("OnEditUserListener",rKZResponseStatus.getMessage());
```

```
    }
  }
});
```

機種変更認証をする（必須項目のみ指定）

機種変更認証(必須項目のみ指定)は RKZClient の authModelChangeCode で行います。

機種変更認証に成功した場合は取得したユーザー情報のユーザーNo と userAccessToken をアラートで表示します。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// 機種変更認証コード発行で発行された認証コードを指定
final String modelChangeCode = “認証コード”;

RKZClient.authModelChange(modelChangeCode, new OnAuthModelChangeListener() {
  @Override
  public void onAuthModelChange(final User user, final RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess()) {
      // 認証成功
      Log.d("OnAuthModelChangeListener", user.getUserAccessToken());
    } else {
      // 認証失敗
      Log.d("OnAuthModelChangeListener",rKZResponseStatus.getStatusCode());
      Log.d("OnAuthModelChangeListener",rKZResponseStatus.getMessage());
    }
  }
});
```

機種変更認証をする（必須項目＋任意項目指定）

機種変更認証(必須項目+任意項目指定)は RKZClient の authModelChangeCode で行います。

機種変更認証に成功した場合は取得したユーザー情報のユーザーNo と userAccessToken をアラートで表示します。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// 機種変更認証コード発行で発行された認証コードを指定
final String modelChangeCode = “認証コード”;
```

```
final String password = “ユーザーが発行した認証パスワード”;

RKZClient.authModelChange(modelChangeCode, password, new OnAuthModelChangeListener() {
  @Override
  public void onAuthModelChange(final User user, final RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess()) {
      // 認証成功
      Log.d("OnAuthModelChangeListener", user.getUserAccessToken());
    } else {
      // 認証失敗
      Log.d("OnAuthModelChangeListener",rKZResponseStatus.getStatusCode());
      Log.d("OnAuthModelChangeListener",rKZResponseStatus.getMessage());
    }
  }
});
```

ユーザーアクセストークンを更新する(1 フェーズコミット)

ユーザーアクセストークン更新(1 フェーズコミット)は RKZClient の updateUserAccessToken で行います。

1 フェーズコミットを利用した場合、Success 時に復帰される userAccessToken がすぐに利用可能な状態となり、旧 userAccessToken は利用不可になります。

取得に成功した場合はコールバックメソッドの第１引数に検索結果のデータが渡されます。
取得失敗の場合は第２引数の isSuccess メソッドが false を返します。

```
// 使用中のユーザーアクセストークンを指定
String userAccessTokene = "ユーザーアクセストークン";

RKZClient.updateUserAccessToken(userAccessToken, new OnUpdateUserAccessTokenListener() {
  @Override
  public void onUpdateUserAccessToken(String newUserAccessToken, RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess()) {
      // 更新成功
      Log.d("OnUpdateUserAccessTokenListener ",newUserAccessToken);
    } else {
      // 更新失敗
      Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getStatusCode());
      Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getMessage());
    }
  }
```

```
});
```

ユーザーアクセストークンを更新する(2 フェーズコミット)

ユーザーアクセストークン更新(2 フェーズコミット)は RKZClient の beginUpdateUserAccessToken で新しいユーザーアクセストークンを仮発行して、commitUpdateUserAccessToken で確定します。

2 フェーズコミットを利用した場合、beginUpdateUserAccessToken にて発行した新しいユーザーアクセストークンは commitUpdateUserAccessToken を呼び出すまで利用できません。

取得に成功した場合はコールバックメソッドの第 1 引数に検索結果のデータが渡されます。
取得失敗の場合は第 2 引数の isSuccess メソッドが false を返します。

```
String userAccessTokene = "ユーザーアクセストークン";      // 使用中のユーザーアクセストークンを指定

RKZClient.beginUpdateUserAccessToken(userAccessToken, new OnUpdateUserAccessTokenListener() {
  @Override
  public void onUpdateUserAccessToken(String newUserAccessToken, RKZResponseStatus rkzResponseStatus) {
    if (rkzResponseStatus.isSuccess()) {
      // 仮発行成功
      RKZClient.commitUpdateUserAccessToken(userAccessToken, new OnUpdateUserAccessTokenListener() {
        @Override
        public void onUpdateUserAccessToken(String newUserAccessToken, RKZResponseStatus rkzResponseStatus) {
          if (rkzResponseStatus.isSuccess()) {
            // 更新成功
            Log.d("OnUpdateUserAccessTokenListener ",newUserAccessToken);
          } else {
            // 更新失敗
            Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getStatusCode());
            Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getMessage());
          }
        }
      });
    } else {
      // 仮登録失敗
      Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getStatusCode());
      Log.d("OnUpdateUserAccessTokenListener ",rKZResponseStatus.getMessage());
    }
  }
});
```

コンタクト管理

コンタクト管理機能を利用する

コンタクト管理機能は、アプリケーションでコンタクト情報を管理する基本的な仕組みを提供します。
このページでは、コンタクト管理機能を利用する実装例を紹介します。

コンタクト情報の一覧を取得する

コンタクトを取得する場合は RKZClient の getContactList で行います。

コンタクトの取得に成功した場合はコールバックリスナの第１引数にコンタクトデータが渡されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// コンタクト種別取得では、必須項目はユーザーアクセストークンです。
// 第２引数に検索条件、第３引数にソート条件を設定できます。
RKZClient.getContactList("APP0001", null, null, new OnGetContactListListener() {
  @Override
  public void onGetContactHistoryList(List<Contact> contactList, RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      for(Contact contact : contactList){
        Log.d("OnGetContactListListener",contact.getContactNo());
        Log.d("OnGetContactListListener",CalendarUtil.getDateText(contact.getContactDate()));
        Log.d("OnGetContactListListener",contact.getContactClassCd());
        Log.d("OnGetContactListListener",contact.getContactMethodClassCd());
        Log.d("OnGetContactListListener",contact.getContactItemNo());
        Log.d("OnGetContactListListener",contact.getEntryNo());
        Log.d("OnGetContactListListener",contact.getStatusCd());
        Log.d("OnGetContactListListener",contact.getPlaceCd());
        Log.d("OnGetContactListListener",StringUtil.parseString(contact.getPoint()));
        Log.d("OnGetContactListListener",contact.getRemarks());
        Log.d("OnGetContactListListener",contact.getDepositNo());
```

```
        Log.d("OnGetContactListListener",contact.getBeaconId());
        Log.d("OnGetContactListListener",contact.getBeaconSpotCd());
        Log.d("OnGetContactListListener",StringUtil.parseString(contact.getRssi()));
        Log.d("OnGetContactListListener",contact.getStampRallyCd());
        Log.d("OnGetContactListListener",contact.getCouponCd());
        Log.d("OnGetContactListListener",StringUtil.parseString(contact.getQuantity()));
        Log.d("OnGetContactListListener",contact.getStampRallyCd());
        Log.d("OnGetContactListListener",contact.getStampRallySpotCd());
      }
    }
  }
});
```

コンタクト情報を登録する

コンタクト情報を登録する場合は RKZClient の addContact で行います。

コンタクト情報の登録に成功した場合はコールバックリスナの第１引数にコンタクトデータが渡されます。
登録に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
//コンタクトデータサンプルを作成
Contact contactHistory = new Contact ();
contact.setContactClassCd("0005");
contact.setContactMethodClassCd("");
contact.setBeaconId("1");
Map<String, Object> attributes = new HashMap<String, Object>();
attributes.put("testfield", "test");
contactHistory.setAttributes(attributes);

// コンタクト履歴登録において必須項目はユーザーアクセストークンです。
RKZClient.addContact ("APP0001", contact, new OnAddContactListener() {
  @Override
  public void onAddContact(RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      Log.d("OnAddContactListener","登録完了");
    } else {
      Log.d("OnAddContactListener",rKZResponseStatus.getStatusCode());
      Log.d("OnAddContactListener",rKZResponseStatus.getMessage());
    }
  }
});
```

お知らせ管理

お知らせ管理機能を利用する

お知らせ管理機能は、アプリケーションでお知らせ情報を管理する基本的な仕組みを提供します。
このページでは、お知らせ管理機能を利用する実装例を紹介します。

すべてのお知らせ情報を取得する（キー未指定）

すべてのお知らせ情報を取得する場合は、RKZClient の getNewsList で行います。

お知らせ取得に成功した場合はコールバックリスナの第１引数にお知らせ情報が渡されます。
取得に失敗した場合は第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。
お気に入り情報（お知らせ）を取得する場合は、拡張属性を指定します。
引数の拡張属性は null を指定しています。

```
//お知らせ取得（公開中のみ）（お知らせ未指定）には最大取得件数が必要になります
// 最大１０件の取得
Integer limit = Integer.valueOf( "10" );

// 第２、３引数には RKZApiSearchCondition と RKZApiSortCondition を指定する。
// 第４引数には NewsExtensionAttribute を指定する（お気に入り情報取得時）。

RKZClient.getInstance().getNewsList( limit, null, null, null, new OnGetReleasedNewsListListener( ) {
    @Override
    public void onGetReleasedNewsListListener(List<News> newsList, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            for ( News news : newsList ) {
                //  お知らせ情報を出力
                Log.d( "OnGetReleasedNewsListListener" , news.getTitle( ) );
                Log.d( "OnGetReleasedNewsListListener" , news.getDescription( ) );
            }
        } else {
            Log.d( "OnGetReleasedNewsListListener" , rkzStatus.getStatusCode( ) );
```

```
            Log.d( "OnGetReleasedNewsListListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

公開中のお知らせ情報を取得する（キー未指定）

公開中のお知らせ情報を取得する場合は、RKZClient の getReleasedNewsList で行います。
お知らせ取得に成功した場合はコールバックリスナの第１引数にお知らせ情報が渡されます。
お知らせ取得に失敗した場合は第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。
お気に入り情報を取得する場合は、拡張属性を指定します。
引数の拡張属性は null を指定しています。

```
// お知らせ取得（公開中のみ）（お知らせ未指定）には最大取得件数が必要になります
// 最大１０件の取得
Integer limit = Integer.valueOf( "10" );

// 第２、３引数には RKZApiSearchCondition と RKZApiSortCondition を指定する。
// 第４引数には NewsExtensionAttribute を指定する。
RKZClient.getInstance().getReleasedNewsList( limit, null, null, null, new OnGetNewsListener( ) {
    @Override
    public void onGetNewsList( List<News> newsList, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            for ( News news : newsList ) {
                // お知らせ情報を出力
                Log.d( "OnGetNewsListListener" , news.getTitle( ) );
                Log.d( "OnGetNewsListListener" , news.getDescription( ) );
            }
        } else {
            Log.d( "OnGetNewsListListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetNewsListListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

お知らせ情報を１レコード取得する（キー指定）

お知らせ情報を１レコード取得する場合は、RKZClient の getNews で行います。

※注意点　readNews で登録したお知らせ既読情報は、取得不可です。

お知らせ取得に成功した場合はコールバックリスナの第１引数にお知らせ情報が渡されます。
失敗した場合は第２引数の isSuccess メソッドが false を返します。

```
// お知らせ ID を指定します
String  newsId = "1";

RKZClient.getInstance().getNews( newsId, new OnGetNewsListener( ) {
    @Override
    public void onGetNews( News news, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            //  お知らせ情報を出力
            Log.d( "OnGetNewsListener" , news.getTitle( ) );
            Log.d( "OnGetNewsListener" , news.getDescription( ) );
        } else {
            Log.d( "OnGetNewsListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetNewsListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

お知らせ既読情報を１レコード取得する（キー指定）

お知らせ既読情報を１レコード取得する場合では、RKZClient の getNewsReadHistory で行います。

※注意点　readNews で登録したお知らせ既読情報は、取得不可です。

お知らせ既読情報取得に成功した場合はコールバックリスナの第１引数にお知らせ既読情報が渡されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// お知らせ既読情報取得（お知らせ指定）には
// お知らせ ID とユーザアクセストークンが必要になります。
String  newsId = "1";
String  userAccessToken  = "userAccessTokenXXXX";
```

```
RKZClient.getInstance().getNewsReadHistory ( newsId, userAccessToken, new OnGetNewsReadHistoryListener( ) {
    @Override
    public void onGetNewsReadHistory( NewsReadHistory newsReadHistory ,
                                      RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
                // お知らせ既読情報を出力
                Log.d( "OnGetNewsReadHistoryListener ", newsReadHistory.getNewsId( ) );
                Log.d( "OnGetNewsReadHistoryListener" , newsReadHistory.getReadDate( ) );
        } else {
            Log.d( "OnGetNewsReadHistoryListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetNewsReadHistoryListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

お知らせ既読情報を複数レコード取得する（キー未指定）

お知らせ既読情報を複数レコード取得する場合は、RKZClient の getNewsReadHistoryList で行います。

※注意点　readNews で登録したお知らせ既読情報は、取得不可です。

お知らせ既読情報取得（お知らせ未指定）の取得に成功した場合は、
コールバックリスナの第１引数にお知らせ既読情報データが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// お知らせ既読情報取得（お知らせ未指定）にはユーザアクセストークンが必要になります
String  userAccessToken  = "userAccessTokenXXXX";

RKZClient.getInstance().getNewsReadHistoryList ( userAccessToken, new OnGetNewsReadHistoryListListener( ) {
    @Override
    public void onGetNewsReadHistoryList( List<NewsReadHistory> newsReadHistoryList,
            RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            for ( NewsReadHistory newsReadHistory : newsReadHistoryList ) {
                // お知らせ既読情報を出力
                Log.d( "OnGetNewsReadHistoryListLitener ",newsReadHistory.getNewsId( ) );
                Log.d( "OnGetNewsReadHistoryListListener", newsReadHistory.getReadDate( ) );
        } else {
            Log.d( "OnGetNewsReadHistoryListLitener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetNewsReadHistoryListLitener" , rkzStatus.getMessage( ) );
        }
    }
}
```

セグメント配信されたお知らせ情報を取得する

BaaS@rakuza の管理者機能にて特定のユーザーに向けたお知らせ配信を行ったとき、引数に渡したユーザーアクセストークンに該当するユーザーに該当するお知らせのみを取得することができます。

セグメント配信されたお知らせ情報を取得する場合は、RKZClient の
getSegmentNewsList で行います。

コールバックリスナの第１引数にお知らせ既読情報データが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

お気に入り情報を取得する場合は、拡張属性を指定します。

```
String userAccessToken = "userAccessTokenXXXX";    // ユーザーアクセストークンを指定します
Integer limit = 10;          // 取得するお知らせの件数を指定します
Boolean onlyMatch = true;    // 自分に関係するお知らせのみを取得するように指定します

RKZClient.getInstance().getSegmentNewsList( limit, userAccessToken, onlyMatch, null, null, null, new OnGetNews
ListListener() {
    @Override
    public void onGetNewsList(List<News> list, RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            for ( NewsData data : list ) {
                // お知らせ既読情報を出力
                Log.d( " OnGetNewsListListener ",data.getNewsId( ) );
        } else {
            Log.d( "OnGetNewsReadHistoryListLitener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetNewsReadHistoryListLitener" , rkzStatus.getMessage( ) );
        }
    }
}
```

お知らせ既読情報を登録する

お知らせ既読情報を登録する場合は、RKZClient の registNewsReadHistory で行います。

※注意点　registNewsReadHistory で登録したお知らせ既読情報は、getNewsReadHistory、または、getNewsReadHistoryList のみで取得可能です。

お知らせ既読情報登録に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// お知らせ既読情報登録にはユーザアクセストークンとお知らせ ID と読んだ既読日時が必要です。
String  userAccessToken  = "userAccessTokenXXXX";
String  newsId = "1";
// 既読日時
Calendar today = Calendar.getInstance();

RKZClient.getInstance().registNewsReadHistory( newsId, userAccessToken, today, new OnRegistNewsReadHistoryList
ener( ) {
    @Override
    public void onRegistNewsReadHistory ( String statusCode, RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            Log.d( "OnRegistNewsReadHistoryListener" , "登録成功" );
        } else {
            Log.d( "OnRegistNewsReadHistoryListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnRegistNewsReadHistoryListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

お知らせ既読情報を登録する

お知らせ既読情報を登録する場合は、RKZClient の readNews で行います。

readNews で登録したお知らせ既読情報は、getNewsReadHistory、または、getNewsReadHistoryList では取得不可です。getNewslist、getSegmentNewsList、getReleasedNewsList、getReleasedSegmentNewsList で取得します。
getNewsList、getReleasedNewsList、getSegmentNewsList、getReleasedSegmentNewsList に、extensionAttribute パラメータを指定することで、未読既読が取得できます。
extensionAttribute にはユーザーアクセストークンを設定します。

```
NewsExtensionAttribute extensionAttribute = new NewsExtensionAttribute();
extetnsionAttribute.setUserAccessToken("userAccessTokenXXXX");
```

お知らせ既読情報登録に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// お知らせ既読情報登録にはユーザアクセストークンとお知らせ ID が必要です。
String  userAccessToken  = "userAccessTokenXXXX";
String  newsId = "1";
```

```
RKZClient.getInstance().readNews( newsId, userAccessToken, new OnReadNewsListener ( ) {
    @Override
    public void onReadNews ( String statusCode, RKZResponseStatus rkzStatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            Log.d( "OnReadNewsListener" , "登録成功" );
        } else {
            Log.d( "OnReadNewsListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnReadNewsListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

プッシュ通知管理

プッシュ通知管理機能を利用する

プッシュ通知管理機能は、アプリケーションを利用するユーザーへプッシュ通知する基本的な仕組みを提供します。
このページでは、プッシュ通知管理機能を利用する実装例を紹介します。

ユーザーのプッシュデバイストークンを登録する

プッシュデバイストークンの設定は RKZClient の registPushDeviceToken で行います。

登録に成功した場合はアラートで「登録に成功しました。」と表示します。
登録に失敗した場合はエラー内容をアラートで表示します。

```
// ユーザーアクセストークンは必須です。
final String userAccessToken = "userAccessTokenXXXX";
// デバイストークンは OS から取得します。
final String deviceToken = "OS から通知されたデバイストークン";

RKZClient.getInstance().registPushDeviceToken(userAccessToken,
                                              deviceToken,
                                              new OnRegistPushDeviceTokenListener() {
  @Override
  public void onRegistPushDeviceToken(final String statusCode, RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            Log.d( "OnRegistPushDeviceTokenListener" , "登録成功" );
        } else {
            Log.d( "OnRegistPushDeviceTokenListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnRegistPushDeviceTokenListener" , rkzStatus.getMessage( ) );
        }
  }
});
```

ユーザーへプッシュ通知する

ユーザーへのプッシュ通知は、管理機能[プッシュ通知管理]カテゴリの各機能から行うことができ

ます。
管理機能[プッシュ通知管理]->[プッシュ通知環境設定]機能より、Android の API キーを設定して利用してください。

アプリケーションでプッシュ通知を受信する

端末でのプッシュ通知の受け取り方法については、Android の受信の仕方を参照してください。

ユーザーのプッシュデバイストークンを削除する

プッシュデバイストークンの削除は RKZClient の clearPushDeviceToken で行います。

登録に成功した場合はアラートで「登録に成功しました。」と表示します。
登録に失敗した場合はエラー内容をアラートで表示します。

```
// ユーザーアクセストークンは必須です。
final String userAccessToken = "userAccessTokenXXXX";

RKZClient.getInstance().clearPushDeviceToken(userAccessToken,
                                             new OnClearPushDeviceTokenListener() {
  @Override
  public void onClearPushDeviceToken(final String statusCode, RKZResponseStatus rkzResponseStatus) {
        if ( rkzStatus.isSuccess( ) ) {
            Log.d( "OnClearPushDeviceTokenListener" , "登録成功" );
        } else {
            Log.d( "OnClearPushDeviceTokenListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnClearPushDeviceTokenListener" , rkzStatus.getMessage( ) );
        }
  }
});
```

ビーコン管理

ビーコン管理機能を利用する

ビーコン管理機能は、アプリケーションでビーコン情報を管理する基本的な仕組みを提供します。
このページでは、ビーコン管理機能を利用する実装例を紹介します。

ビーコンを複数レコード取得する

ビーコンを複数取得する場合は、RKZClient の getBeaconList で行います。

ビーコン取得に成功した場合はコールバックリスナの第１引数にビーコンデータが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// ビーコン取得では必須項目はありません
// 第１引数に検索条件、第２引数にソート条件を指定できます
RKZClient.getBeaconList( null, null, new OnGetBeaconListListener( ) {
    @Override
    public void onGetBeaconList( List<Beacon> beaconList ,
                                 RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            for ( Beacon beacon : beaconList ) {
                //  ビーコン情報を出力
                Log.d( "OnGetBeaconListListener" , beacon.getCode( ) );
                Log.d( "OnGetBeaconListListener" , beacon.getName( ) );
                Log.d( "OnGetBeaconListListener" , beacon.getShortName( ) );
                Log.d( "OnGetBeaconListListener" , beacon.getBeaconId( ) );
                Log.d( "OnGetBeaconListListener" , beacon.getBeaconTypeCd( ) );
                Log.d( "OnGetBeaconListListener" , StringUtil.parseString(beacon.getMajor( ) ) );
                Log.d( "OnGetBeaconListListener" , StringUtil.parseString(beacon.getMinor( ) ) );
            }
        } else {
            Log.d( "OnGetBeaconListListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetBeaconListListener" , rkzStatus.getMessage( ) );
```

```
        }
    }
}
```

スポット情報を複数レコード取得する

スポットを複数取得する場合は、RKZClient の getSpotList で行います。

スポット取得に成功した場合はコールバックリスナの第１引数にスポットデータが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// ビーコン取得では必須項目はありません
// 第１引数に検索条件、第２引数にソート条件を指定できます
RKZClient.getSpotList( null, null, new OnGetSpotListListener() {
      @Override
      public void onGetSpotList(final List<Spot> list, final RKZResponseStatus rkzResponseStatus) {
        if( rkzStatus.isSuccess( ) ) {
          for ( Spot spot : list ) {
            //  スポット情報を出力
            Log.d( " OnGetSpotListListener " , spot.getCode( ) );
            Log.d( " OnGetSpotListListener " , spot.getName( ) );
            Log.d( " OnGetSpotListListener " , spot.getShortName( ) );
          }
        } else {
            Log.d( "OnGetBeaconListListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetBeaconListListener" , rkzStatus.getMessage( ) );
        }
      }
    }
);
```

クーポン管理

クーポン管理機能を利用する

クーポン管理機能は、アプリケーションでクーポン情報を管理する基本的な仕組みを提供します。
このページでは、クーポン管理機能を利用する実装例を紹介します。

クーポンを複数レコード取得する

クーポンを複数取得する場合は、RKZClient の getCouponList で行います。

クーポン取得に成功した場合はコールバックリスナの第１引数にクーポンデータが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// クーポン取得（coupon 未指定）では必須項目はありません
// 第１引数に検索条件、第２引数にソート条件を指定できます
RKZClient.getCouponList( null, null, new OnGetCouponListListener( ) {
    @Override
    public void onGetCouponList( List<Coupon> couopnList ,
                                 RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            for ( Coupon coupon : couponList ) {
                // クーポン情報を出力
                Log.d( "OnGetCouponListListener", coupon.getCode() );
                Log.d( "OnGetCouponListListener", coupon.getName() );
                Log.d( "OnGetCouponListListener", CalendarUtil.getDateText( coupon.getPossibleFromDte() ) );
            }
        } else {
            Log.d( "OnGetCouponListListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetCouponListListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

クーポンを１レコード取得する

クーポンを１レコード取得する場合は、RKZClient の getCoupon で行います。

クーポン取得に成功した場合はコールバックリスナの第１引数クーポンデータが渡されます。
取得に失敗した場合は第２引数の isSuccess メソッドが false を返します。

```
// クーポンコード指定します
String  couponCode = "0001";

RKZClient.getCoupon( couponCode, new OnGetCouponListener( ) {
    @Override
    public void onGetCoupon( Coupon coupon , RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            // クーポン情報を出力
            Log.d( "OnGetCouponListener" , coupon.getCode( ) );
            Log.d( "OnGetCouponListener" , coupon.getName( ) );
            Log.d( "OnGetCouponListener" , CalendarUtil.getDateText (
                    coupon.getPossibleFromDte( ) );
        } else {
            Log.d( "OnGetCouponListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetCouponListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

クーポンを交換する

クーポンを交換する場合は、RKZClient の exchangeCoupon で行います。

クーポン交換に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
クーポン交換に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// クーポン交換では、ユーザアクセストークンとクーポンコードと交換枚数が必要です
String  userAccessToken  = "userAccessTokenXXXX";
String coupontCode = "0005";
Integer quantity = Integer.valueOf( "1" );
```

```
RKZClient.exchangeCoupon( userAccessToken, couponCode, quantity, new OnExchangeCouponListener( ) {
    @Override
    public void onExchangeCouponListener( String statusCode, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            Log.d( "OnExchangeCouponListener" , "クーポン交換に成功しました。" );
        } else {
            Log.d( "OnExchangeCouponListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnExchangeCouponListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

マイクーポンを複数レコード取得する

マイクーポンを複数取得する場合は、RKZClient の getMyCouponList で行います。

マイクーポン取得に成功した場合はコールバックリスナの第１引数にマイクーポンデータが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// マイクーポン取得（myCoupon 未指定）ではユーザアクセストークンが必要です
String userAccessToken = "userAccessTokenXXXX";

// 第２引数に検索条件、第３引数にソート条件を指定できます
RKZClient.getMyCouponList( userAccessToken, null, null, new OnGetMyCouponListListener( ) {
    @Override
    public void onGetMyCouponList( List<MyCoupon> myCouponList, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            for ( MyCoupon myCoupon : myCouponList ) {
                Log.d( "OnGetMyCouponListListener" , myCoupon.getCode( ) );
                Log.d( "OnGetMyCouponListListener" , myCoupon.getCouponName( ) );
                Log.d( "OnGetMyCouponListListener" , myCoupon.getCouponCd( ) );
            }
        } else {
            Log.d( "OnGetMyCouponListListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetMyCouponListListener" , rkzStatus.getMessage( ) );
```

```
        }
    }
}
```

マイクーポンを１レコード取得する

マイクーポンを１レコード取得する場合は、RKZClient の getMyCoupon で行います。

マイクーポン取得に成功した場合はコールバックリスナの第１引数にマイクーポンデータが渡されます。
取得に失敗した場合は第２引数の isSuccess メソッドが false を返します。

```
// マイクーポン取得（myCoupon 指定）ではユーザアクセストークン及びマイクーポンコードが必要です
String  userAccessToken  = "userAccessTokenXXXX";
String myCouponCode = "10";

RKZClient.getMyCoupon( userAccessToken, myCouponCode, new OnGetMyCouponListener( ) {
    @Override
    public void onGetMyCoupon( MyCoupon myCoupon , RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            Log.d( "OnGetMyCouponListener" , myCoupon.getCode( ) );
            Log.d( "OnGetMyCouponListener" , myCoupon.getCouponName( ) );
            Log.d( "OnGetMyCouponListener" , myCoupon.getCouponCd( );
        } else {
            Log.d( "OnGetMyCouponListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetMyCouponListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

クーポンを利用する

クーポンを利用する場合は、RKZClient の useMyCoupon で行います。

クーポン利用に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
クーポン利用に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// クーポン利用では、ユーザアクセストークンとマイクーポンデータが必要です
```

```
String  userAccessToken  = "userAccessTokenXXXX";
MyCoupon myCoupon = new MyCoupon( );
// マイクーポンデータにはマイクーポンコードとクーポンコードが設定されている必要がある
myCoupon.setCode("31");
myCoupon.setCouponcd("0005");

RKZClient.useMyCoupon( userAccessToken, myCoupon, new OnUseMyCouponListener( ) {
    @Override
    public void onUseMyCoupon( String statusCode , RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            Log.d( "OnUseMyCouponListener" , "クーポン利用に成功しました。" );
        } else {
            Log.d( "OnExchangeCouponListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnExchangeCouponListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

ポイント管理

ポイント管理機能を利用する

ポイント管理機能は、アプリケーションでユーザーが保持するポイント情報を管理する基本的な仕組みを提供します。
このページでは、ポイント管理機能を利用する実装例を紹介します。

ユーザーのポイント情報を取得する

ユーザーが保持しているポイント情報を取得する場合は RKZClient の getPoint で行います。

取得に成功した場合はコールバックメソッドの第１引数にポイント情報として渡されます。
取得に失敗した場合は第２引数の isSuccess メソッドが false を返します。

```
//  ポイント取得にはポイントを知りたいアプリ利用者のユーザアクセストークンが必要です
String  userAccessToken  = "userAccessTokenXXXX";

RKZClient.getPoint( userAccessToken, new OnGetPointListener( ) {
    @Override
    public void onGetPoint( Point point, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            Log.d( "OnGetPointListener" , point.getPoint() );
        } else {
            Log.d( "OnGetPointListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetPointListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

ユーザーのポイント数を加算・減算する

ユーザーの保持しているポイント情報を加算・減算する場合は RKZClient の addPoint で行います。

ポイントの加算・減算処理に成功した場合はコールバックメソッドの第１引数に処理後のポイント情報が渡されます。

失敗した場合は第２引数の isSuccess メソッドが false を返します。

```
// ポイント加算減算にはアプリ利用者のユーザアクセストークンと加算減算するポイント数と日付が必要です
String  userAccessToken  = "userAccessTokenXXXX";
// １００ポイント追加させる
Integer pointNum  = Integer.valueOf("100");
// ポイント計算を行った日付
Calendar today = Calendar.getInstance();

RKZClient.addPoint( userAccessToken,  pointNum,  today,  new OnGetPointListener( ) {
    @Override
    public void onGetPoint( Point point, RKZResponseStatus rkzStatus ) {
        if( rkzStatus.isSuccess( ) ) {
            Log.d( "OnGetPointListener" , Integer.toString(point.getPoint() ) );
        } else {
            Log.d( "OnGetPointListener" , rkzStatus.getStatusCode( ) );
            Log.d( "OnGetPointListener" , rkzStatus.getMessage( ) );
        }
    }
}
```

アプリ管理

アプリ管理機能を利用する

アプリ管理機能は、アプリケーションの設定を管理する基本的な仕組みを提供します。
このページでは、アプリ管理機能を利用する実装例を紹介します。

アプリケーション設定情報を取得する

アプリケーション基本設定情報の取得は RKZClient の getApplicationSettingData で行います。

取得に成功した場合は、コールバックリスナの第１引数に取得データが渡されます。
取得に失敗した場合は、第２引数の isSuccess メソッドが false を返します。

```
RKZClient.getApplicationSettingData( OnGetApplicationSettingDataListener() {
    @Override
    public void onGetApplicationSettingData( ApplicationConfig appConfig,
                                             RKZResponseStatus rkzSatus ) {
        if ( rkzStatus.isSuccess( ) ) {
            // 成功時
            Log.d( "OnGetApplicationSettingData", appConfig.getName( ) );
            Log.d( "OnGetApplicationSettingData", appConfig.getVersionAndroid( ) );
        } else {
            Log.d( "OnGetApplicationSettingData", rkzStatus.getStatusCode( ) );
            Log.d( "OnGetApplicationSettingData", rkzStatus.getMessage( ) );
        }
    }
}
```

スタンプラリー管理

スタンプラリー管理機能を利用する

スタンプラリー管理機能は、アプリケーションでスタンプラリー情報を管理する基本的な仕組みを提供します。
このページでは、スタンプラリー管理機能を利用する実装例を紹介します。

スタンプラリー情報（開催中）を一覧取得する

スタンプラリー一覧を取得する場合は、RKZClient の getStampRallyList で行います。

スタンプラリー一覧取得に成功した場合はコールバックリスナの第 1 引数にスタンプラリーデータリストが渡されます。
取得に失敗した場合はコールバックリスナの第 2 引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
//スタンプラリー一覧取得では必須項目は有りません。
//第１引数に検索条件、第２引数に、ソート条件を指定できます。
RKZClient.getStampRallyList(null, null, new OnGetStampRallyListListener() {
  @Override
  public void onGetStampRallyList(List<StampRally> stamprallyDataList, RKZResponseStatus rKZResponseStatus) {

    if(rKZResponseStatus.isSuccess()){
      for(StampRally stampRally : stamprallyDataList){
        // スタンプラリー情報を出力
        Log.d("OnGetStampRallyListListener",stampRally.getCode());
        Log.d("OnGetStampRallyListListener",stampRally.getName());
        Log.d("OnGetStampRallyListListener",stampRally.getShortName());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyDetail());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImage());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImageUrl());
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyStartDate()));
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyEndDate()));
      }
```

```
    }
  }
});
```

スタンプラリー情報（全取得）を一覧取得する

スタンプラリーを全件取得する場合は、RKZClient の getAllStampRallyList で行います。

スタンプラリー取得に成功した場合はコールバックリスナの第１引数にスタンプラリーデータリストが渡されます。
取得に失敗した場合は lp-るバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
//スタンプラリー一覧取得では必須項目は有りません。
//第１引数に検索条件、第２引数に、ソート条件を指定できます。
RKZClient.getStampRallyList(null, null, new OnGetStampRallyListListener() {
  @Override
  public void onGetStampRallyList(List<StampRally> stamprallyDataList, RKZResponseStatus rKZResponseStatus) {

    if(rKZResponseStatus.isSuccess()){
      for(StampRally stampRally : stamprallyDataList){
        // スタンプラリー情報を出力
        Log.d("OnGetStampRallyListListener",stampRally.getCode());
        Log.d("OnGetStampRallyListListener",stampRally.getName());
        Log.d("OnGetStampRallyListListener",stampRally.getShortName());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyDetail());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImage());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImageUrl());
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyStartDate()));
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyEndDate()));
      }
    }
  }
});
```

スタンプラリースポット情報（必須条件なし）を一覧取得する

スタンプラリースポット情報一覧取得（必須条件なし）は RKZClient の getStampRallySpotList で行います。

取得に成功した場合は、取得したレコードのコードと名称をアラートで表示します。
取得に失敗した場合はエラー内容をアラートで表示します。

引数の検索条件とソート条件は未指定です。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// スタンプラリー一覧取得では必須項目は有りません。
// 第１引数に検索条件、第２引数に、ソート条件を指定できます。
RKZClient.getAllStampRallyList(null, null, new OnGetStampRallyListListener() {
  @Override
  public void onGetStampRallyList(List<StampRally> stamprallyDataList, RKZResponseStatus rKZResponseStatus) {

    if(rKZResponseStatus.isSuccess()){
      for(StampRally stampRally : stamprallyDataList){
        // スタンプラリー情報を出力
        Log.d("OnGetStampRallyListListener",stampRally.getCode());
        Log.d("OnGetStampRallyListListener",stampRally.getName());
        Log.d("OnGetStampRallyListListener",stampRally.getShortName());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyDetail());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImage());
        Log.d("OnGetStampRallyListListener",stampRally.getStampRallyImageUrl());
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyStartDate()));
        Log.d("OnGetStampRallyListListener",CalendarUtil.getDateText(stampRally.getStampRallyEndDate()));
      }
    }
  }
});
```

スタンプラリースポット情報（スタンプラリー指定）を一覧取得する

スタンプラリースポット一覧をスタンプラリーID で指定して取得する場合は、RKZClient の getStampRallySpotListByStampRallyId で行います。

スタンプラリースポット一覧取得に成功した場合はコールバックリスナの第１引数にスタンプラリースポット一覧データリストが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// スタンプラリースポット一覧取得では必須項目はスタンプラリーID です
// 第１引数に検索条件、第２引数にソート条件を設定できます。
RKZClient.getStampRallySpotListByStampRallyId("0001",null, null,
      new OnGetStampRallySpotListListener() {
  @Override
  public void onGetStampRallySpotList(List<StampRallySpot> stamprallyspotDataList,
                                      RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      for(StampRallySpot stampRallySpot : stamprallyspotDataList){
        // スタンプラリースポット情報を出力
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getCode());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getStampRallyCd());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getStampRallyName());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getName());
        Log.d("OnGetStampRallySpotListListener",StringUtil.parseString(stampRallySpot.getSortNo()));
      }
    }
  }
});
```

スタンプラリースポット情報（スポット指定）を一覧取得する

スタンプラリースポット一覧をスポットで指定する場合、RKZClient の getStampRallySpotListBySpotId で行います。

スタンプラリースポット一覧取得に成功した場合はコールバックリスナの第１引数にスタンプラリースポット一覧データリストが渡されます。
取得に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

引数の検索条件とソート条件は null を指定しています。
検索条件、ソート条件の設定方法については
データ管理 ＞ 複数レコード取得する（キー未指定）＞ 検索条件について
データ管理 ＞ 複数レコード取得する（キー未指定）＞ ソート条件について
を参照してください。

```
// スタンプラリースポット一覧取得では必須項目はスポット ID です。
// 第１引数に検索条件、第２引数にソート条件を設定できます。
RKZClient.getStampRallySpotListBySpotId("0001",null, null,
                                                    new OnGetStampRallySpotListListener() {
  @Override
  public void onGetStampRallySpotList(List<StampRallySpot> stamprallyspotDataList,
                                      RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      for(StampRallySpot stampRallySpot : stamprallyspotDataList){
        //スタンプラリースポット情報を出力
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getCode());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getStampRallyCd());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getStampRallyName());
        Log.d("OnGetStampRallySpotListListener",stampRallySpot.getName());
        Log.d("OnGetStampRallySpotListListener",StringUtil.parseString(stampRallySpot.getSortNo()));
      }
    }
  }
});
```

スタンプコンプリートを登録する

スタンプラリー情報のスタンプコンプリート登録は RKZClient の stampComplete で行います。

登録に成功したか失敗したかを取得することができます。

登録に成功した場合はアラートで「スタンプをコンプリートしました。」と表示します。
登録に失敗した場合はエラー内容をアラートで表示します。

```
// スタンプコンプリートでは必須項目はユーザーアクセストークン、スタンプラリーID です。
RKZClient.stampComplete("APP0001", "0001", new OnStampCompleteListener() {
  @Override
  public void onStampComplete(RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      Log.d("OnStampCompleteListener","登録成功");
    } else {
      Log.d("OnStampCompleteListener",rKZResponseStatus.getStatusCode());
      Log.d("OnStampCompleteListener",rKZResponseStatus.getMessage());
    }
  }
});
```

取得したスタンプを登録する

取得したスタンプを登録する場合は、RKZClient の addMyStamp で行います。

取得したスタンプの登録に成功した場合はコールバックリスナの第１引数に取得したスタンプデータが渡されます。
登録に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
// 取得スタンプ登録では必須項目はユーザーアクセストークン、スタンプラリーID,スタンプラリースポット ID です。
RKZClient.addMyStamp("APP0001", "0001", "0001", new OnAddMyStampListener() {
  @Override
  public void onAddMyStamp(RKZResponseStatus rKZResponseStatus) {
    if (rKZResponseStatus.isSuccess()){
      Log.d("OnAddMyStampListener","登録成功");
    } else {
      Log.d("OnAddMyStampListener",rKZResponseStatus.getStatusCode());
      Log.d("OnAddMyStampListener",rKZResponseStatus.getMessage());
    }
  }
});
```

スタンプ取得履歴を取得する

スタンプ取得履歴の取得は RKZClient の getMyStampHistoryList で行います。

取得に成功した場合は取得したレコードのコンタクト種別コードとポイントをアラートで表示します。
取得失敗の場合はエラー内容をアラートで表示します。

引数の検索条件とソート条件は未指定です。
検索条件、ソート条件の設定方法については
データ管理 > 複数レコード取得する（キー未指定） > 検索条件について
データ管理 > 複数レコード取得する（キー未指定） > ソート条件について
を参照してください。

```
// 取得スタンプ履歴取得では、必須項目はユーザーアクセストークンです。
RKZClient.getMyStampHistoryList("APP0001", null, null, new OnGetAcquisitionStateOfStampListener() {
  @Override
  public void onGetAcquisitionStateOfStamp(List<MyStampHistory> myStampHistoryList,
          RKZResponseStatus rKZResponseStatus) {
    if(rKZResponseStatus.isSuccess()){
      for(MyStampHistory myStampHistory : myStampHistoryList){
        Log.d("OnGetAcquisitionStateOfStampListener",myStampHistory.getContactClassCd());
        Log.d("OnGetAcquisitionStateOfStampListener",myStampHistory.getStampRallyCd());
        Log.d("OnGetAcquisitionStateOfStampListener",myStampHistory.getStampRallyName());
        Log.d("OnGetAcquisitionStateOfStampListener",myStampHistory.getStampRallySpotCd());
        Log.d("OnGetAcquisitionStateOfStampListener",myStampHistory.getStampRallySpotName());
        Log.d("OnGetAcquisitionStateOfStampListener",CalendarUtil.getDateText((myStampHistory.getContactDate
()));
      }
    }
  }
});
```

お気に入り管理

お気に入り管理機能を利用する

お気に入り管理機能は、アプリケーションでお気に入り情報を管理する基本的な仕組みを提供します。
このページでは、お気に入り管理機能を利用する実装例を紹介します。

オブジェクトデータをお気に入りに登録する

オブジェクトデータをお気に入りに登録する場合は、RKZClient の addFavoriteToObjecttData で行います。

お気に入り情報登録に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
お気に入り情報登録に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
//お気に入り登録では、必須項目は、オブジェクトデータとユーザーアクセストークンです。
// RKZObjectData を作成し、登録に最低限必要なデータを設定します
RKZObjectData rKZData = new RKZObjectData( );
rKZData.setObjectId("object1" );
rKZData.setCode("0001" );

//ユーザーアクセストークンを指定
String userAccessTokene = "ユーザーアクセストークン";

RKZClient.getInstance().addFavoriteToObjectData ( rKZData, userAccessTokene, new OnAddFavoriteListener() {
  @Override
  public void onAddFavorite(RKZResponseStatus rKZResponseStatus) {
    if ( rkzStatus.isSuccess( ) ) {
        Log.d( "OnAddFavoriteListener" , "登録成功" );
    } else {
        Log.d( "OnAddFavoriteListener" , rkzStatus.getStatusCode( ) );
        Log.d( "OnAddFavoriteListener" , rkzStatus.getMessage( ) );
    }
  }
});
```

オブジェクトデータのお気に入りを削除する

オブジェクトデータのお気に入りを削除する場合は、RKZClient の deleteFavoriteToObjectData で行います。

お気に入り情報削除に成功した場合はコールバックリスナの第１引数に"1001"が渡されます。
お気に入り情報削除に失敗した場合はコールバックリスナの第２引数の isSuccess メソッドが false を返します。

```
//お気に入り削除では、必須項目は、オブジェクトデータとユーザーアクセストークンです。
// RKZObjectData を作成し、削除に最低限必要なデータを設定します
RKZObjectData rKZData = new RKZObjectData( );
rKZData.setObjectId("object1" );
rKZData.setCode("0001" );

//ユーザーアクセストークンを指定
String userAccessTokene = "ユーザーアクセストークン";

RKZClient.getInstance().deleteFavoriteToObjectData( rKZData, userAccessTokene, new OnDeleteFavoriteListener()
{
  @Override
  public void onAddFavorite(RKZResponseStatus rKZResponseStatus) {
    if ( rkzStatus.isSuccess( ) ) {
        Log.d( "OnDeleteFavoriteListener" , "削除成功" );
    } else {
        Log.d( "OnDeleteFavoriteListener" , rkzStatus.getStatusCode( ) );
        Log.d( "OnDeleteFavoriteListener" , rkzStatus.getMessage( ) );
    }
  }
});
```

タイムアウトの制御

API のタイムアウトを制御する

全ての API 呼び出し時に、タイムアウトを指定することで API のレスポンスが未応答の場合の処理を制御する機能を提供します。
このページでは、スタンプラリー管理機能を利用する実装例を紹介します。

全ての API で共通のタイムアウト時間を設定する

全ての API で共通のタイムアウト時間を設定するには RKZClient の setDefaultTimeouto で行います。

Blocks により処理結果を返却します。
処理時間が指定した時間以内で完了した場合は " rKZResponseStatus.isSuccess" が True, 処理時間が指定した時間以上経過した場合は False となります。

```
// デフォルトタイムアウト時間の設定
RKZClient.getInstance().setDefaultTimeout(10);
RKZClient.getInstance().getSpotList(null, null, new OnGetSpotListListener() {
    @Override
    public void onGetSpotList(List<Spot> spotList, RKZResponseStatus rKZResponseStatus) {
        if (rKZResponseStatus.isSuccess()) {
            Log.i("baasatrakuza", "SUCCESS! ");
        } else {
            Log.e("baasatrakuza", "ERROR! ");
        }
    }
});
```

API 個別にタイムアウト時間を設定する

特定の API にのみ有効なタイムアウト時間を設定するには RKZClient の setTimeouto で行います。

setTimeout メソッドを呼び出すと、指定したタイムアウト時間でタイムアウトする RKZClient インスタンスが新しく生成されます。生成されたインスタンスから呼び出される API は全て setTimeout で指定した時間がタイムアウトとして有効になります。

Blocks により処理結果を返却します。
処理時間が指定した時間以内で完了した場合は "responseStatus.isSuccess" が YES , 処理時間が指定した時間以上経過した場合は NO となります。

```
// タイムアウト時間の設定
RKZClient.getInstance().setDefaultTimeout(10);
RKZClient.getInstance().getSpotList(null, null, new OnGetSpotListListener() {
    @Override
    public void onGetSpotList(List<Spot> spotList, RKZResponseStatus rKZResponseStatus) {
        if (rKZResponseStatus.isSuccess()) {
            Log.i("baasatrakuza", "SUCCESS! ");
        } else {
            // 10秒以上かかる場合はエラーとなる
            Log.e("baasatrakuza", "ERROR! ");
        }
    }
});
```

※注意点 setDefaultTimeout と setTimeout どちらも指定した場合は setTimeout にて指定した時間がタイムアウト値として有効になります。

更新履歴

版数	日付	更新内容
第 1 版	2017/01/27	◆ Ver2.0.0 対応版　初版。
第 2 版	2018/02/2	◆ Ver2.1.0 対応版
第 3 版	2019/06/06	◆ Ver2.2.0 対応版