BaaS@rakuza Cordovaプラグイン リファレンス

ver 1

ピープルソフトウェア株式会社

目次

クイックスタート

BaaS@rakuza Cordovaプラグイン の利用方法

このページでは、BaaS@rakuza Cordovaプラグイン 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。
・Monaca
・OS Windows 7 64bit

Monaca

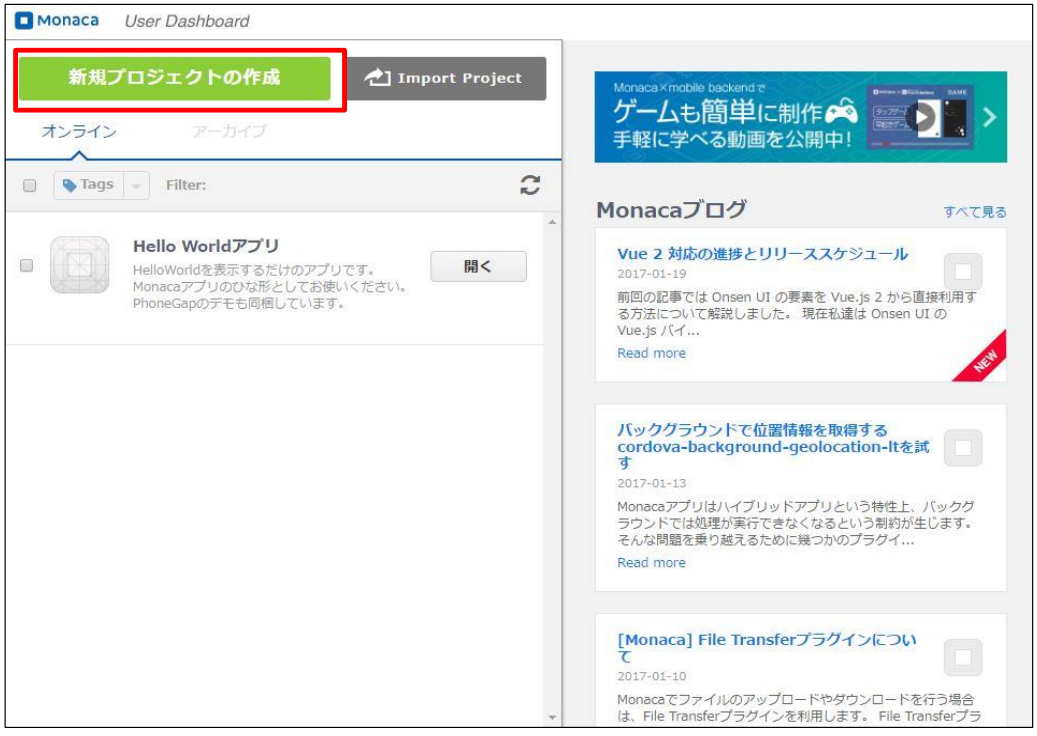
以下のURLからMonacaアカウントを作成してください。アカウント登録は無料です。
アカウント登録が完了すれば、すぐにプロジェクトを作成できます。

https://ja.monaca.io/register/start.html

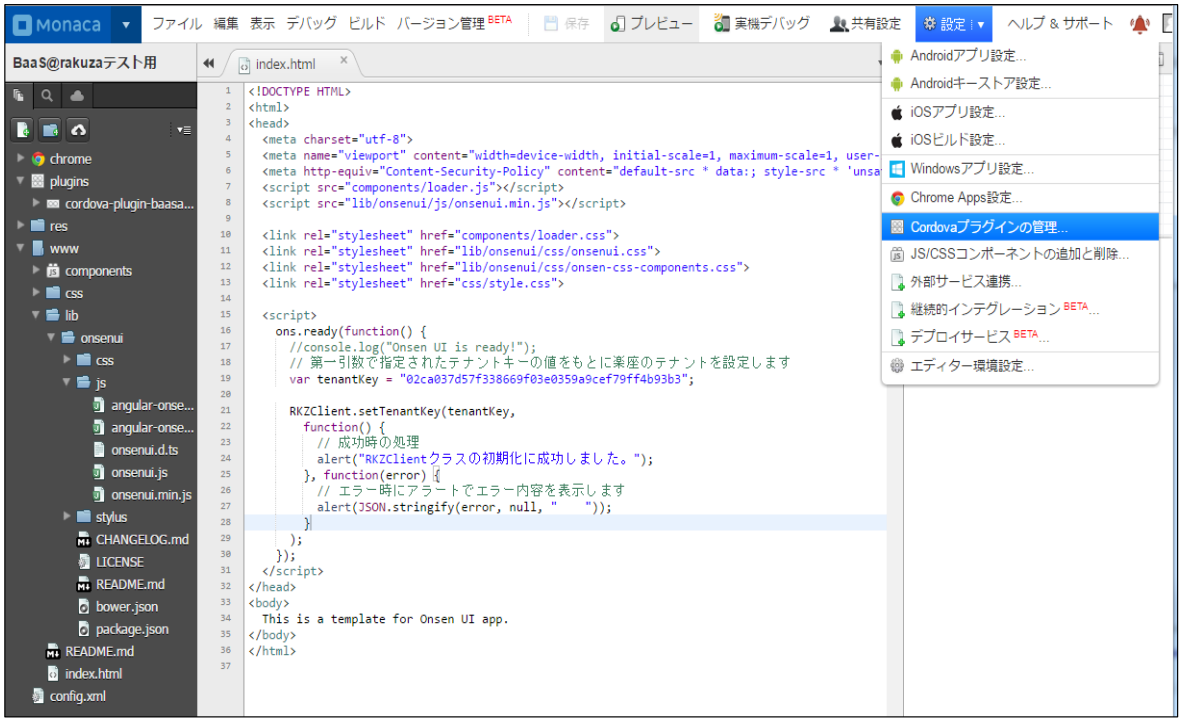
プロジェクトへの設定

BaaS@rakuza Cordovaプラグイン をプロジェクトに追加する

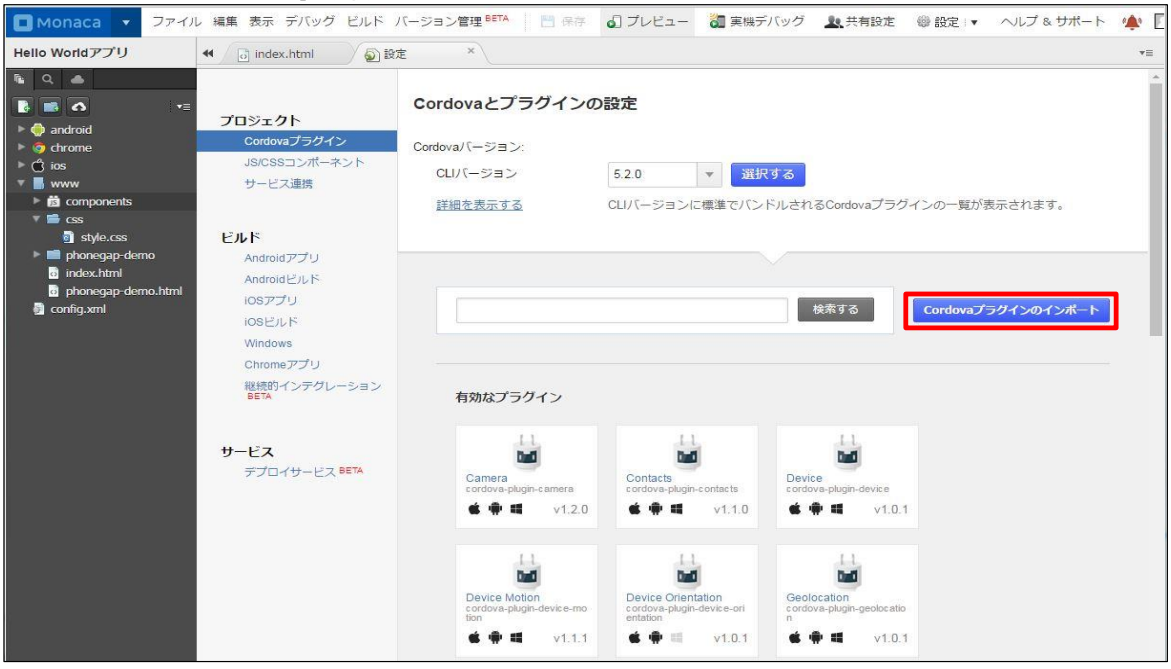
Monacaアカウントを作成しログインすると以下のようなダッシュボード画面が表示されます。
「新規プロジェクトの作成」を選択し、作成するテンプレートを選択します。
テンプレートは　OnsenUI　＞　Onsen UI V2 JS Minimum　が最低限のテンプレートになります。



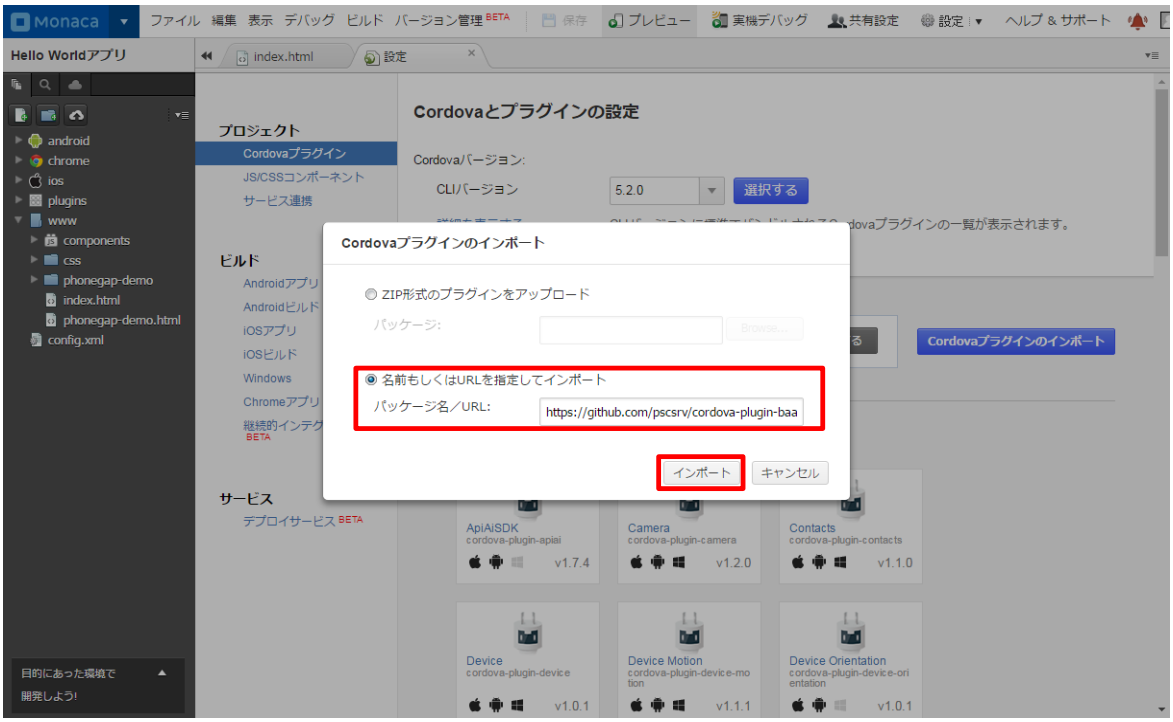
作成したプロジェクトの開くと、以下のような開発画面が表示されます。
メニューから「設定」→「Cordovaプラグインの管理」を選択します。



Cordovaとプラグインの設定　画面からBaaS@rakuzaプラグインの追加を行います。
「Cordovaプラグインのインポート」を選択します。



申込時にご連絡させていただきましたCordvaプラグインdownload用 URLを入力し、「インポート」を選択します。



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 を利用する際には、RKZClientの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 を使用するうえで重要なRKZClientクラスの初期化を説明します。
BaaS@rakuza ではRKZClientクラスの初期化は最初に呼び出される画面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。

・ www / lib / index.html

index.html を開き、<script>タグの初期処理内に以下のコードを追加します。

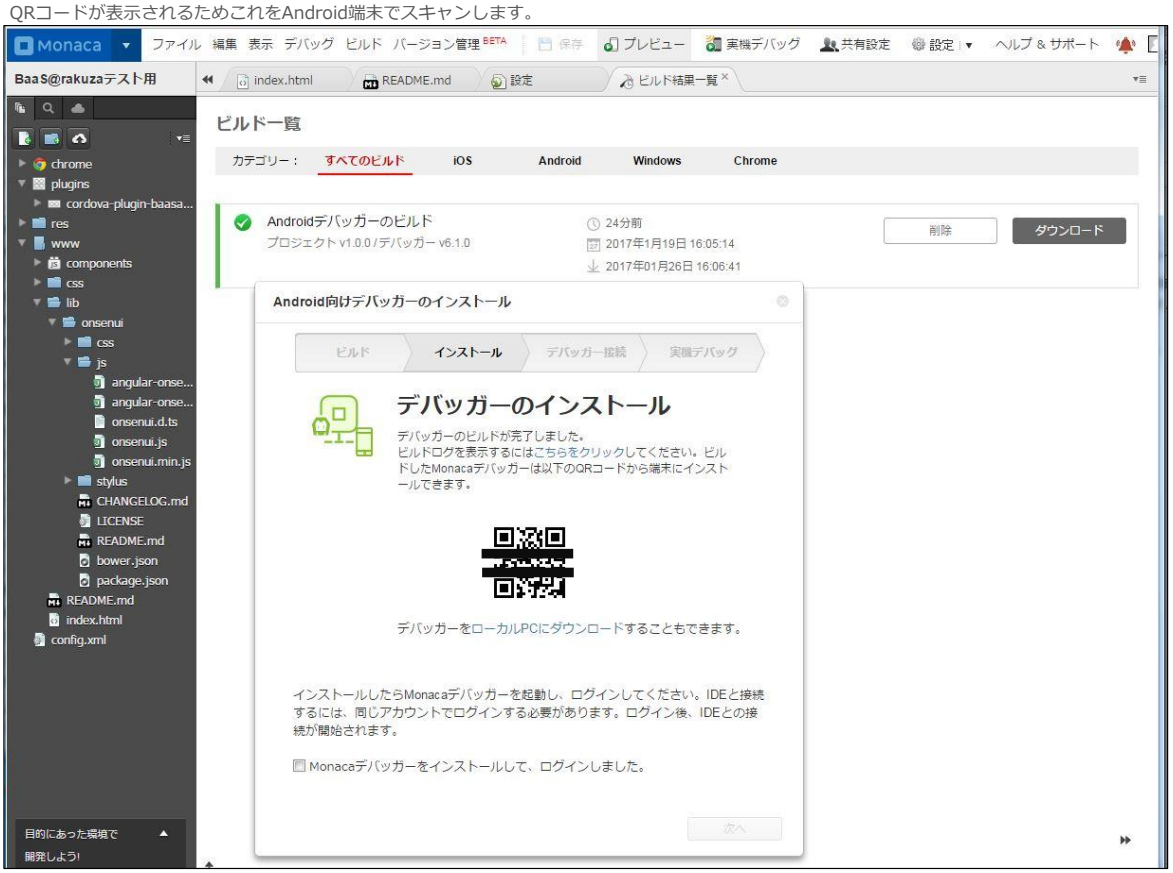
```
// 第一引数で指定されたテナントキーの値をもとに楽座のテナントを設定します
var tenantKey = "配布したテナントキー";

RKZClient.setTenantKey(tenantKey,
  function() {
    // 成功時の処理
    alert("RKZClientクラスの初期化に成功しました。");
  }, function(error) {
    // エラー時にアラートでエラー内容を表示します
    alert(JSON.stringify(error, null, "    "));
  }
);
```

動作確認(Android)

Android端末に作成したアプリケーションをインストールして動作確認を行います。
メニューから「設定」→「Androidアプリ設定」を選択
アプリケーション情報の「アプリケーション名」と「パッケージ名」を変更します。
※「アプリケーション名」はアプリアイコンの下などに表示されるアプリ名です。
　「パッケージ名」は一意になるように設定してください。

メニューから「ビルド」→「Androidアプリのビルド」を選択 （確認のポップアップが出た場合はOKを押します。）
「デバッガーのビルド」を選択し「ビルド開始」ボタンを押します。
ビルドが完了したら「ダウンロードボタン」を押します。
QRコードが表示されるためこれをAndroid端末でスキャンします。



読み取ったURLにアクセスし、画面の指示に従ってファイルをダウンロードし、アプリケーションをインストールします。
インストールしたアプリケーションを起動し、"RKZClientクラスの初期化に成功しました。"と表示されれば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。
ブラウザ上でエラーが出る場合は誤字脱字などを確認してください。

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 -> 検索条件について」　「データ管理 -> 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。

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します
var targetRecord = [];

// オブジェクト内の項目名称がpriceに指定されている値の"1000"から"5000"のオブジェクトデータを取得します
targetRecord[targetRecord.length] = '1000';
targetRecord[targetRecord.length] = '5000';

// 複数指定の物を探し出す命令を示すRKZSearchCondition.inを指定し、
// 検索する項目名称を引数で指定します
var searchCondition = [
    RKZSearchCondition.betweenInclude("price", targetRecord)
];

// ソート条件データを作成します
// オブジェクト内のコードを降順でソートする場合
// RKZSortConditionクラスのコンストラクタで
// 降順を示すRKZSortCondition.descを指定し、
// 引数にソートしたい項目名称を指定します
var sortCondition = [
    RKZSortCondition.desc("code")
];

// 検索条件、ソート条件を指定しないで取得します
// 取得したいオブジェクトIDは必須項目です
var objectId = "drink_menu";

// 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引数に指定します
RKZClient.getDataList(objectId, searchCondition, sortCondition,
    function(datas) {
        // 成功時に取得したコードと名称をアラートでソート順に表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].code + "," + datas[i].name);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

検索条件について

各種情報の取得APIで指定する検索条件を設定・操作するオブジェクトです。
APIでの検索条件の指定は、各種の検索条件設定メソッドで返されるRKZSearchConditionオブジェクトを指定します。
配列で複数の検索条件を設定した場合、AND条件の指定となります。

定数名	条件
RKZSearchCondition.equal	検索値と一致
RKZSeachCondition.notEqual	検索値と不一致
RKZSearchCondition.in	検索値のいずれかに一致
RKZSearchCondition.notIn	検索値のいずれにも不一致
RKZSearchCondition.lessThanInclude	指定した検索値以下
RKZSearchCondition.greaterThanInclude	検索した検索値以上
RKZSearchCondition.betweenInclude	検索した検索値の範囲内（検索値含む）
RKZSearchCondition.betweenExclude	指定した検索値の範囲内（検索値を含まない）
RKZSearchCondition.likeBefore	検索値に前方一致
RKZSearchCondition.likeAfter	検索値に後方一致
RKZSearchCondition.likeBoth	検索値に部分一致
RKZSearchCondition.likeOr	検索値と一致（チェックボックス専用）

ソート条件について

各種情報の取得APIで指定するソート条件を設定・操作するオブジェクトです。
APIでのソート条件の指定は、ソート条件設定メソッドで返されるRKZSortConditionオブジェクトを指定します。
配列で複数のソート条件を設定した場合、配列の先頭から優先順位が決定されます。

定数名	条件
RKZSortCondition.asc	昇順
RKZSortCondition.desc	降順

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。

取得に成功した場合は取得したレコードのコード、名称、短縮名称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。

```
// 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とオブジェクトのコードが必要になります
var objectId = "drink_menu";
var code = "0001";

RKZClient.getData(objectId, code,
    function(data) {
        // 成功時にアラートで取得したデータを表示します
       alert(data.code + "," +  data.name + "," + data.short_name);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

オブジェクトデータを登録する

オブジェクトデータ登録は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。

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

var objectId = "drink_menu";
var name = "オレンジジュース";

rKZData.object_id = objectId;
rKZData.name = name;

// オブジェクトデータを登録します（RKZClient変数は初期化済みとする）
RKZClient.addData(rKZData,
    function() {
        // 成功時にアラートでメッセージを表示します
        alert("登録に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

オブジェクトデータを編集する

オブジェクトデータ編集は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アラートで「編集に成功しました。」と表示します。
編集に失敗した場合はエラー内容をアラートで表示します。

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

// 必須項目は編集できません
var objectId = "drink_menu";
var code = "0004";
var name = "オレンジジュース";
var shortName = "オレンジ";

rKZData.object_id = objectId;
rKZData.code = code;
rKZData.name = name;
rKZData.short_name = shortName;

// オブジェクトデータを編集します（RKZClient変数は初期化済みとする）
RKZClient.editData(rKZData,
    function() {
        // 成功時にアラートでメッセージを表示します
        alert("編集に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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登録されたuserAccessTokenとユーザーNoをアラートで表示します。
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ください。

```
// 登録するユーザー情報を作成します
var userData = {};

// ユーザー情報を登録します（RKZClient変数は初期化済みとする）
RKZClient.registUser(userData,
    function(userData) {
        // 成功時にアラートでuserAccessTokenとユーザーNoを表示します
        alert(userData.user_access_token + "," + userData.user_no);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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取得したユーザー情報のユーザーNo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。

```
// ユーザー情報取得ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// ユーザー情報を取得します（RKZClient変数は初期化済みとする）
RKZClient.getUser(userAccessToken,
    function(data) {
        // 成功時にアラートでuserAccessToken、ユーザーNoを表示します
        alert(data.user_access_token + "," + data.user_no);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

ユーザー情報を編集する

ユーザー情報の編集はRKZClientの editUser で行います。

ユーザーのメールアドレスを編集します。
ユーザー情報の編集に成功した場合は取得したユーザー情報のユーザーNoをアラートで表示します。
編集に失敗した場合はエラー内容をアラートで表示します。

```
// 編集するユーザー情報を作成します
var userData = {};

// ユーザー情報編集ではuserAccessTokenが必須項目です
userData.userAccessToken = "userAccessTokenXXXX";

// メールアドレスを設定します
userData.attributes.mail_address_1 = "XXXXXX@XXXXX.co.jp";

// ユーザー情報を編集します（RKZClient変数は初期化済みとする）
RKZClient.editUser(userData
    function(data) {
        // 成功時にアラートでuserAccessToken、ユーザーNoを表示します
        alert(data.user_access_token + "," + data.user_no);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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をアラートで表示します。
発行に失敗した場合はエラー内容をアラートで表示します。

```
// 機種変更認証コード発行(必須項目のみ指定)ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// 機種変更認証コードを発行します（RKZClient変数は初期化済みとする）
RKZClient.registModelChangeCode(userAccessToken,
    function(data) {
        // 成功時にアラートで認証コード、有効期限を表示します
        alert(data.model_change_code + "," + data.limit_date);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

機種変更認証コードを発行する(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をアラートで表示します。
発行に失敗した場合はエラー内容をアラートで表示します。

```
// 機種変更認証コード発行(必須項目のみ指定)ではuserAccessTokenが必須項目です
var userAccessToken = userData.user_access_token;

// 任意項目(パスワード、桁数、有効時間)の設定を行います
var password = "password";
var limitCode = 4;
var limitMinute = 10;

// 機種変更認証コードを発行します（RKZClient変数は初期化済みとする）
RKZClient.registModelChangeCode(userAccessToken, password, limitCode, limitMinute,
    function(data) {
        // 成功時にアラートで認証コード、有効期限を表示します
        alert(data.model_change_code + "," + data.limit_date);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。
認証に失敗した場合はエラー内容をアラートで表示します。

```
// 機種変更認証では認証コードは必須項目です
var modelChangeCode = "認証コード";

// 機種変更認証をします（RKZClient変数は初期化済みとする）
RKZClient.authModelChangeCode(modelChangeCode,
    function(data) {
        // 成功時にアラートでユーザーNo、userAccessTokenを表示します
        alert(data.user_no + "," + data.user_access_token);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。
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。

```
// 機種変更認証では認証コードは必須項目です
var modelChangeCode = "認証コード";

//  任意項目(パスワード)の設定を行います
var password = "パスワード";

// 機種変更認証をします（RKZClient変数は初期化済みとする）
RKZClient.authModelChangeCode(modelChangeCode, password,
    function(data) {
        // 成功時にアラートでユーザーNo、userAccessTokenを表示します
        alert(data.user_no + "," + data.user_access_token);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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

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 -> 検索条件について」 「データ管理 -> ソート条件について」
を参照してください。

コンタクト情報の一覧取得はRKZClientの getContactList 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とポイント増減数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。

```
// 検索条件データを作成します
var targetRecord = [];

// コンタクト種別が"0001"か"0002"のオブジェクトデータを取得します
targetRecord[targetRecord.length] = '0001';
targetRecord[targetRecord.length] = '0002';

// 複数指定の物を探し出す命令を示すRKZSearchCondition.inを指定し、
// 検索する項目名称を引数で指定します
var searchCondition = [
    RKZSearchCondition.in("contact_class_cd", targetRecord)
];

// ソート条件データを空で作成します
var sortCondition = [];

// コンタクト情報一覧取得ではuserAccessTokenは必須項目です
var userAccessToken = "userAccessTokenXXXX";

// 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.getContactList(userAccessToken, searchCondition, sortCondition,
    function(datas) {
        // 成功時に取得したコンタクト種別とポイント増減数をアラートで表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].contact_class_cd + "," + datas[i].point);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

コンタクト情報を登録する

コンタクト情報の登録はRKZClientの addContact 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「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。

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

// コンタクト情報登録ではコンタクト種別とコンタクト方法は必須項目です
var contactCC = "0001";
var contactMCC = "0001";

// 付与するポイント増減数を設定します
var point = 10;

rKZData.contact_class_cd = contactCC;
rKZData.contact_method_class_cd = contactMCC;
rKZData.point = point;

// コンタクト情報登録ではuserAccessTokenは必要項目です
var userAccessToken = "userAccessTokenXXXX";

// コンタクト情報を登録します（RKZClient変数は初期化済みとする）
RKZClient.addContact(userAccessToken, rKZData,
    function() {
        // 成功時にアラートでメッセージを表示します
        alert("登録に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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、取得したお知らせ情報のタイトルとカテゴリー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（コード未指定）」
を参照してください。

```
// お知らせ取得（キー未指定）では最大取得件数が必須項目です
// 最大１０件を取得します
var limit = 10;

// 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.getNewsList(limit, [], [],
    function(datas) {
        // 成功時にアラートでタイトルとカテゴリーを表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].title + "," + datas[i].category);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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、取得したお知らせ情報のタイトルとカテゴリー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（コード未指定）」
を参照してください。

```
// お知らせ取得（公開中のみ）（キー未指定）では最大取得件数が必須項目です
// 最大１０件を取得します
var limit = 10;

// 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.getReleasedNewsList(limit, [], [],
    function(datas) {
        // 成功時にアラートでタイトルとカテゴリーを表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].title + "," + datas[i].category);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

お知らせ取得に成功した場合は、取得したお知らせ情報のタイトルとカテゴリーをアラートで表示します。
失敗した場合はエラー内容をアラートで表示します。

```
// お知らせIDを指定します
var params = {news_id :"1"};

// お知らせ情報を1レコード取得します
RKZClient.getNews( params,
    function(data) {
        // 成功時にアラートでタイトルとカテゴリーを表示します
        alert(data.title + "," + data.category);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

お知らせ既読情報取得に成功した場合は、取得したお知らせ既読情報のお知らせIDを表示します。
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。

```
// お知らせ既読情報取得（キー指定）では
// お知らせIDとuserAccessTokenが必須項目です
var params = {news_id :"0001"};
var userAccessToken = "userAccessTokenXXXX";

// お知らせ既読情報を1レコード取得します（RKZClient変数は初期化済みとする）
RKZClient.getNewsReadHistory(params, userAccessToken,
    function(data) {
        // 成功時にアラートでお知らせIDを表示します
        alert(data.news_id);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

お知らせ既読情報取得（キー未指定）の取得に成功した場合は
取得したお知らせ既読情報のお知らせID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。

```
// お知らせ既読情報取得（キー未指定）ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// お知らせ既読情報を複数レコード取得します（RKZClient変数は初期化済みとする）
RKZClient.getNewsReadHistoryList(userAccessToken,
    function(datas) {
        // 成功時にアラートでお知らせIDを表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].news_id);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

お知らせ既読情報を登録する

お知らせ既読情報の登録はRKZClientの registNewsReadHistory 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。

お知らせID＝1の内容を既読登録します。
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。

```
// 既読にする対象のお知らせ情報を設定します
var params = {
    news_id: "1",
    tenant_id: "XXXXX",
    read_date: new Date()};

// お知らせ既読情報の登録ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// オブジェクトデータを登録します（RKZClient変数は初期化済みとする）
RKZClient.registNewsReadHistory(params, userAccessToken,
    function(StatusCode) {
        // 成功時にアラートでメッセージを表示します
        alert("登録に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

```
// プッシュデバイストークンの登録ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// プッシュデバイストークンの登録ではdeviceTokenが必須項目です
var deviceToken = XXXX取得したプッシュデバイストークンXXXX;

// プッシュデバイストークンを登録します（RKZClient変数は初期化済みとする）
RKZClient.registPushDeviceToken(userAccessToken, deviceToken
    function(statusCode) {
        // 成功時にアラートでメッセージを表示します
        alert("登録に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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、iOSのプッシュ通知の証明書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取方法については、Cordova等、他のプラグインを参照してください。

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、取得した順にビーコン端末名称とビーコンID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（コード未指定）」
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([], [],
    function(datas) {
        // 成功時にアラートでビーコン端末名称とビーコンIDを表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].name + "," + datas[i].beacon_id);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

スポット情報を複数レコード取得する

スポット情報の取得はRKZClientの getSpotList で行います。

取得に成功した場合は取得した順にコードとスポット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（コード未指定）」
を参照してください。

```
// スポットリスト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([], [],
    function(datas) {
        // 成功時に取得したコードとスポット名称をアラートでソート順に表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].code + "," + datas[i].name);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error,null,"    "));
    }
);
```

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、取得した順にクーポンのクーポン名称とコード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（コード未指定）」
を参照してください。

```
// クーポン複数レコード取得（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([], [],
    function(datas) {
        // 成功時にアラートでクーポン名称とコードを表示します
        for (var i = 0; i < datas.length; i++) {
            // アラートで取得した値を表示
            alert(datas[i].name + "," + datas[i].code);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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、取得したクーポンのクーポン名称とコード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。

```
// コードを指定します
var couponCode = "0001";

// クーポンを1レコード取得します
RKZClient.getCoupon( couponCode,
    function(data) {
        // 成功時にアラートでクーポン名称とコードを表示します
        alert(data.name + "," + data.code);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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「クーポン交換に成功しました。」とアラートで表示します。
クーポン交換に失敗した場合はエラー内容をアラートで表示します。

```
// クーポン交換ではuserAccessTokenとクーポンコードと交換枚数が必須項目です
var userAccessToken = "userAccessTokenXXXX";
var couponCode = "0002";
var quantity = 1;

// クーポンを交換します（RKZClient変数は初期化済みとする）
RKZClient.exchangeCoupon(userAccessToken, couponCode, quantity,
    function() {
        // 成功時にアラートでメッセージを表示します
        alert("クーポン交換に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

マイクーポンの複数取得に成功した場合は、取得した順にクーポンと残りのクーポン数量をアラートで表示します。
マイクーポンの複数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（コード未指定）」
を参照してください。

```
// マイクーポンの複数レコード取得ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// 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.getMyCouponList(userAccessToken, [], [],
    function(datas) {
        //  成功時に取得したクーポンと残りのクーポン数量をアラートで表示します
        for (var i = 0; i < datas.length; i++) {
            alert(datas[i].coupon_cd + "," + datas[i].quantity);
        }
    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

マイクーポンの1レコード取得に成功した場合は、取得したクーポンと残りのクーポン数量をアラートで表示します。
マイクーポンの1レコード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。

```
// マイクーポンの1レコード取得ではuserAccessTokenとクーポンコードが必須項目です
var userAccessToken = "userAccessTokenXXXX";
var couponCode = "0002";

// マイクーポンを1レコード取得します（RKZClient変数は初期化済みとする）
RKZClient.getMyCoupon(userAccessToken, couponCode,
    function(data) {
        //  成功時に取得したクーポンと残りのクーポン数量をアラートで表示します
        alert(data.coupon_cd + "," + data.quantity);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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アラートで「クーポン利用に成功しました。」と表示します。
クーポン利用に失敗した場合はエラー内容をアラートで表示します。

```
// クーポン利用ではuserAccessTokenとマイクーポン情報が必須項目です
var userAccessToken = "userAccessTokenXXXX";
var myCoupon = {};

// マイクーポン情報にはクーポンコードと
// マイクーポンコードが設定されている必要があります
myCoupon.coupon_cd = "0002";
myCoupon.code = "0002";

// クーポンを利用します（RKZClient変数は初期化済みとする）
RKZClient.useMyCoupon(userAccessToken, myCoupon,
    function() {
        //  成功時にアラートでメッセージを表示します
        alert("クーポン利用に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

ポイント情報取得に成功した場合は、ポイント増減数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。

```
// ユーザー情報取得ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// ユーザー情報を取得します（RKZClient変数は初期化済みとする）
RKZClient.getPoint(userAccessToken,
    function(data) {
        // 成功時にアラートでポイント増減数を表示します
        alert(data.point);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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。

加算・減算に成功した場合はアラートで「更新に成功しました。」と表示します。
失敗した場合はエラー内容をアラートで表示します。

```
// ユーザー情報取得ではuserAccessTokenが必須項目です
var userAccessToken = "userAccessTokenXXXX";

// 加算・減算するポイント数を設定します
var point = 10;

// システム日付をポイント取得日に設定します
var getDate = new Date();

// ポイント数を加算・減算します（RKZClient変数は初期化済みとする）
RKZClient.addPointt(userAccessToken, point, getDate,
    function(data) {
        // 成功時にアラートでメッセージを表示します
        alert("更新に成功しました。");

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```

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取得したレコードのアプリケーション名と略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。

```
// アプリケーション設定情報では必須項目はありません
RKZClient.getApplicationSettingData(
    function(data) {
        // 成功時にアラートでアプリケーション名と略名称を表示します
        alert(data.name + "," + data.short_name);

    }, function(error) {
        // エラー時にアラートでエラー内容を表示します
        alert(JSON.stringify(error, null, "    "));
    }
);
```