BaaS@rakuza

SDK リファレンスマニュアル

Cordova プラグイン版

Ver1.2.0

第 3 版

PEOPLE SOFTWARE

内容

クイックスタート

BaaS@kuraza 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

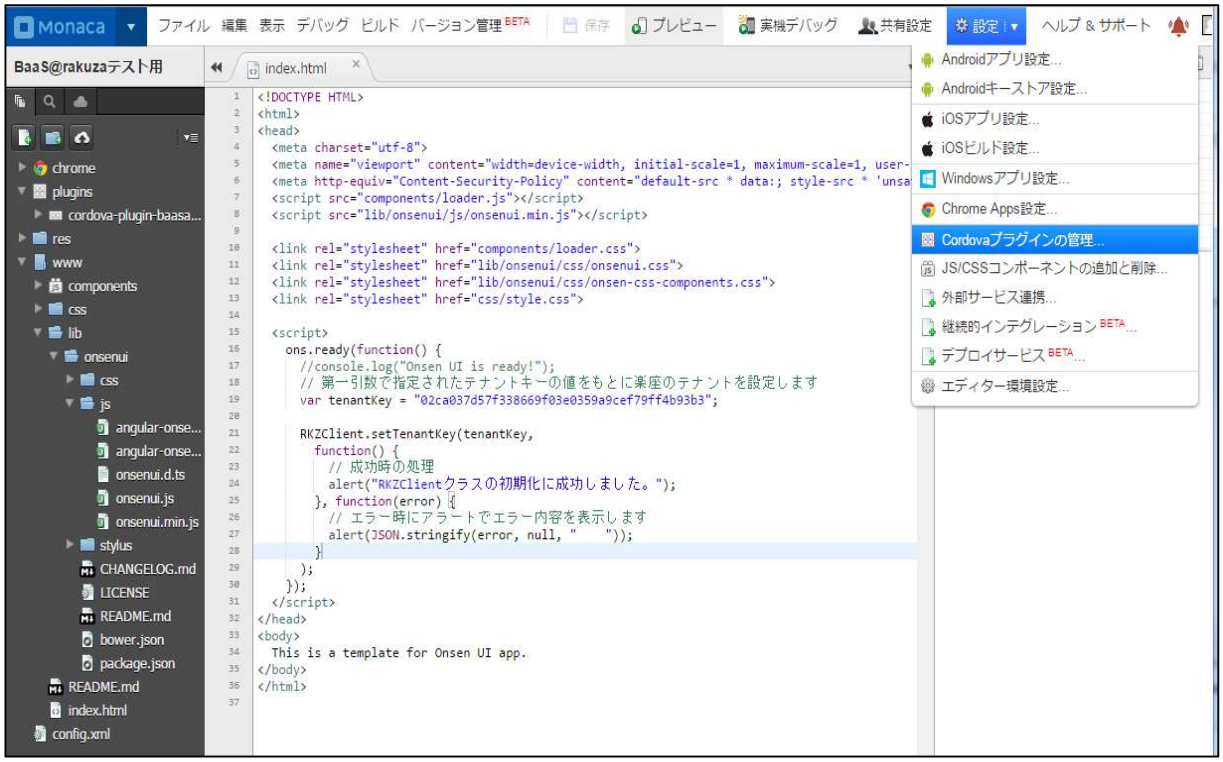
プロジェクトへの設定

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。

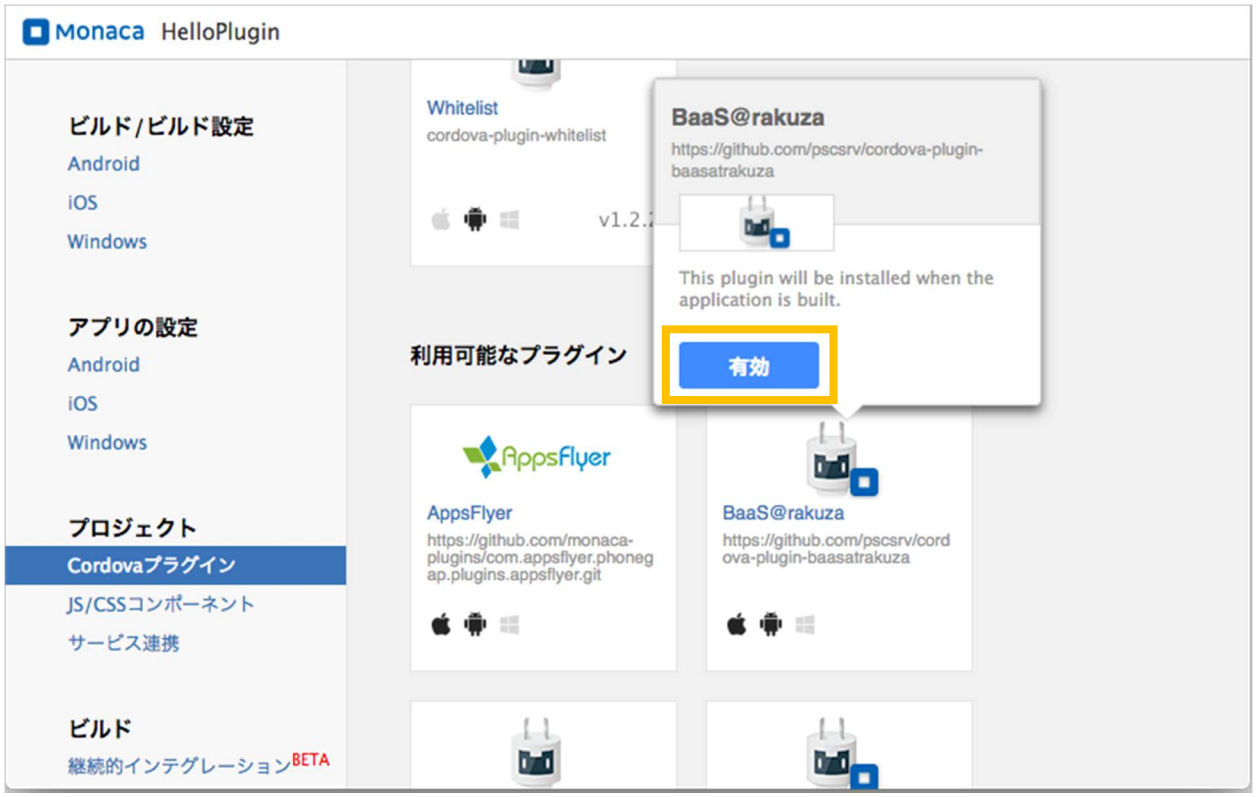
テンプレートは　OnsenUI　＞　Onsen UI V2 JS Minimum　が最低限のテンプレートになります。

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。

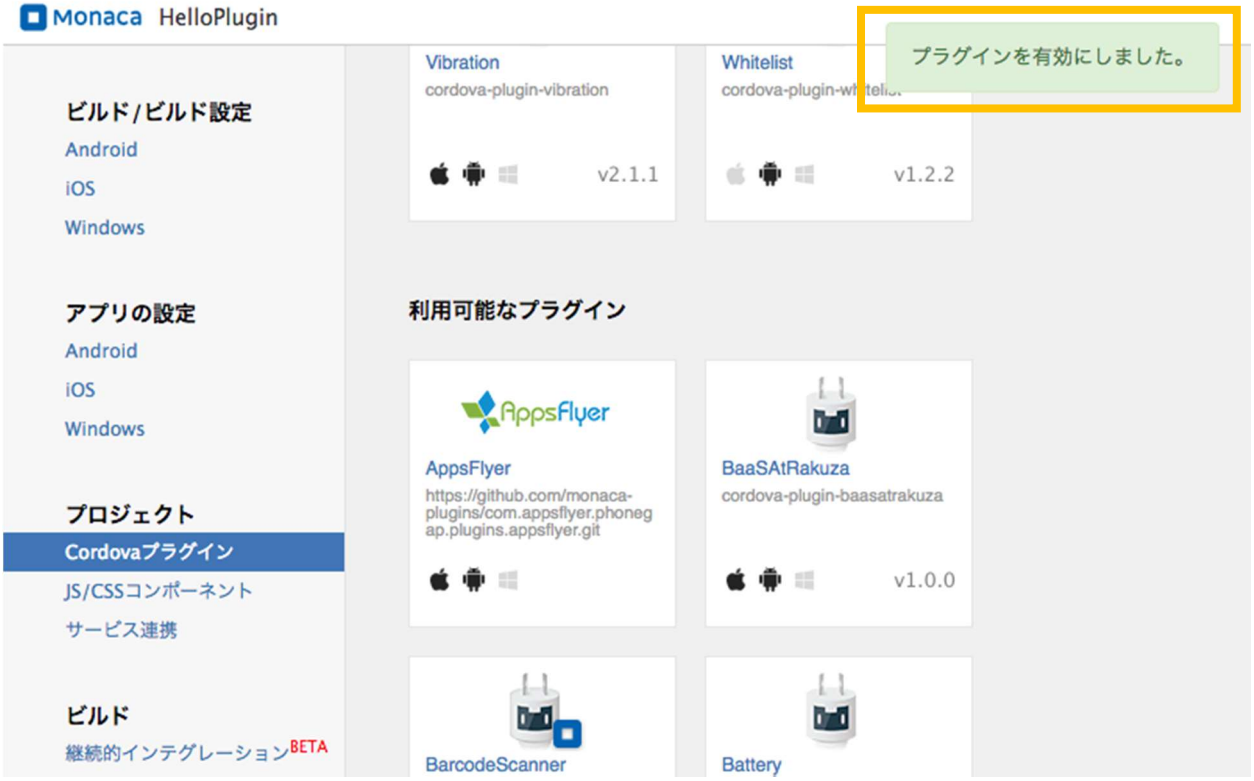


Cordova プラグインの設定画面から BaaS@rakuza プラグインの追加を行います。
「利用可能なプラグイン」の欄に表示されている BaaS@rakuza にカーソルを合わせます。

吹き出し画面が表示されるので、表示された吹き出し内の「有効」ボタンをクリックします。



画面右上に「プラグインを有効にしました。」と表示されれば追加は完了です。



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 を初期化する

ここでは BaaS@rakuza を使用するうえで重要な RKZClient の初期化を説明します。
BaaS@rakuza では RKZClient 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。

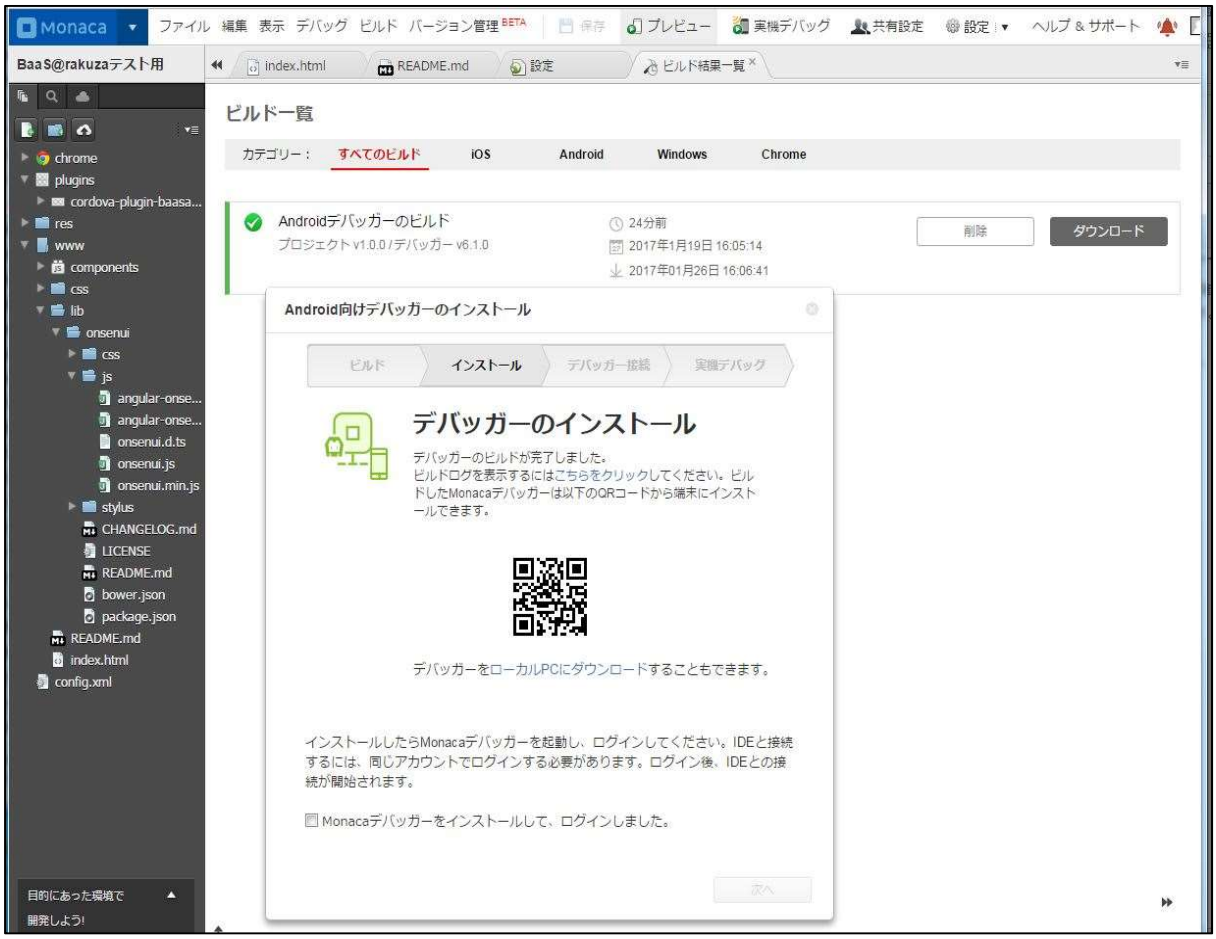
```
// 第一引数で指定されたテナントキーの値をもとに楽座のテナントを設定します
var tenantKey = "配布したテナントキー";

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 ＞ 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
// 拡張属性を使用する場合は、
// 第４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

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.equal	検索値のいずれかに該当する
RKZSeachCondition.notEqual	検索値のいずれにも該当しない
RKZSearchCondition.in	検索値と一致
RKZSearchCondition.notIn	検索値と一致一致しない
RKZSearchCondition.lessThanInclude	検索値に前方一致する
RKZSearchCondition.greaterThanInclude	検索値に後方一致する
RKZSearchCondition.betweenInclude	検索値に部分一致する
RKZSearchCondition.betweenExclude	検索した検索値の範囲内（検索値含む）
RKZSearchCondition.likeBefore	指定した検索値の範囲内（検索値を含まない）
RKZSearchCondition.likeAfter	指定した検索値以上
RKZSearchCondition.likeBoth	検索した検索値以下
RKZSearchCondition.likeOr	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
RKZSearchCondition.WithFavorite. 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
RKZSearchCondition.WithFavorite.All	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
RKZSearchCondition.ReadedNews.AlreadyRead	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
RKZSearchCondition.ReadedNews.NoRead	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
RKZSearchCondition.ReadedNews.All	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.withFavorite.updateDate.asc	お気に入り登録された日付を昇順でソートします。
RKZSortCondition.withFavorite.updateDate.desc	お気に入り登録された日付を降順でソートします。
RKZSortCondition.favoriteCount.asc	お気に入り登録された件数を昇順でソートします。
RKZSortCondition.favoriteCount.desc	お気に入り登録された件数を降順でソートします。

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。

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, "    "));
    }
);
```

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

```
// 削除対象となるオブジェクト ID を指定します。(必須)
var objectId = " drink_menu ";
// 削除するデータの条件を指定します。(条件なしの場合は全データ削除となります)
var searchConditions = null;

RKZClient.deleteData(objectId,
    searchConditions,
    function() {
        // 成功時にアラートでメッセージを表示します
        alert( "削除に成功しました。" );
    }, 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。

```
var objectId = "drink_menu";
var searchCondition = [];
var sortCondition = [];

// extensionAttribute 引数を設定
var extensionAttribute = {
    user_access_token: "user_access_token",
    show_favorite: true,     // お気に入り情報を復帰する
    show_favorite_summary: true     // お気に入り件数を復帰する
};

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].attributes.sys_favorite.is_favorite);
            alert(datas[i].code + "," + datas[i].attributes.sys_favorite.favorite_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, "    "));
    }
);
```

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下さい。

```
// 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

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。

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。

```
// 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 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。

```
// 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;

// 拡張属性を指定します。
var extensionAttribute = {
    "user_access_token": helper.userAccessToken,
    "show_favorite": true,
    "show_favorite_summary": true
};

// 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, [], [], extensionAttribute,
    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）

すべて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 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。

```
// 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;

// 拡張属性を指定します。
var extensionAttribute = {
    "user_access_token": helper.userAccessToken,
    "show_favorite": true,
    "show_favorite_summary": true
};

// 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, [], [], extensionAttribute,
    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。

※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、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。

※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（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。

※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。

```
// 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, "    "));
    }
);
```

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 readNews 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。

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。

```
var extensionAttrribute = {
    user_access_token: "userAccessTokenXXXX"
};
```

```
// 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"};

// 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.readNews(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け取り方法については、iOS 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。

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

// プッシュデバイストークンを削除します（RKZClient 変数は初期化済みとする）
RKZClient.clearPushDeviceToken( 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, "    "));
    }
);
```

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 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([], [],
    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 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 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 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, "    "));
    }
);
```

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

スタンプラリー情報一覧取得（開催中）は RKZClient の getStampRallyList 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 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。

```
// スタンプラリー情報の一覧取得（開催中）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.getStampRallyList([], [],
    function(datas) {
        // 成功時に取得したコードと名称をアラートで表示します
        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, "    "));
    }
);
```

スタンプラリー情報（全取得）を一覧取得する

スタンプラリー情報一覧取得（全取得）は RKZClient の getAllStampRallyList 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 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。

```
// スタンプラリー情報の一覧取得(全取得)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.getAllStampRallyList([], [],
    function(datas) {
        // 成功時に取得したコードと名称をアラートで表示します
        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,"    "));
    }
);
```

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。

```
// スタンプラリースポット情報の一覧取得(全取得)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.getAllStampRallyList([], [],
    function(datas) {
        // 成功時に取得したコードと名称をアラートで表示します
        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,"    "));
    }
);
```

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情報一覧取得（スタンプラリー指定）は RKZClient の getStampRallySpotListByStampRallyId 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 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。

```
// スタンプラリーオブジェクトのコードを指定します
var stampRallyId = "0002";

// 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.getStampRallySpotListByStampRallyId(stampRallyId, [], [],
    function(datas) {
        // 成功時に取得したコードと名称をアラートで表示します
        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,"    "));
    }
);
```

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情報一覧取得（スポット指定）は RKZClient の getStampRallySpotListBySpotId 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 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オブジェクトのコードを指定します
var spotId = "0001";

// 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.getStampRallySpotListBySpotId(spotId, [], [],
    function(datas) {
        // 成功時に取得したコードと名称をアラートで表示します
        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,"    "));
    }
);
```

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登録では userAccessToken とスタンプラリーオブジェクトのコードが必須項目です
var userAccessToken = "userAccessTokenXXXX";
var stampRallyId = "0001";

// スタンプコンプリートを登録する（RKZClient 変数は初期化済みとする）
RKZClient.stampComplete(userAccessToken, stampRallyId,
    function(datas) {
        // 成功時にアラートでメッセージを表示します
        alert("スタンプをコンプリートしました。");
    }, 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, "    "));
    }
);
```

取得したスタンプを登録する

スタンプラリーで取得したスタンプの登録は RKZClient の addMyStamp 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「取得したスタンプを登録しました。」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。

```
// 取得スタンプ登録では userAccessToken とスタンプラリーオブジェクトのコードと
// スタンプラリースポットオブジェクトのコードが必須項目です
var userAccessToken = "userAccessTokenXXXX";
var stampRallyId = "0001";
var spotId = "0001"

// 取得したスタンプを登録する（RKZClient 変数は初期化済みとする）
RKZClient.addMyStamp(userAccessToken, stampRallyId, spotId,
    function() {
```

```
        // 成功時にアラートでメッセージを表示します
        alert("取得したスタンプを登録しました。");

    }, 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, "    "));
    }
);
```

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。

```
// スタンプ取得履歴取得では userAccessToken は必須項目です
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.getMyStampHistoryList(userAccessToken, [], [],
    function(datas) {
        // 成功時に取得した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
        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, "    "));
    }
);
```

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のお気に入りを登録する

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 addFavoriteToObjectData で行います。

登録に成功した場合はアラートで「お気に入りを登録しました。」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。

```
// お気に入り登録ではオブジェクトデータと userAccessToken が必須項目です
var objectData = {
    object_id: "object1",
    code: "0001"
};
var userAccessToken = "userAccessTokenXXXX";

// お気に入りを登録する（RKZClient 変数は初期化済みとする）
RKZClient.addFavoriteToObjectData( objectData, userAccessToken,
    function(statusCode) {
        // 成功時にアラートでメッセージを表示します
        alert("お気に入りを登録しました。");

    }, 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。

削除に成功した場合はアラートで「お気に入りを削除しました。」と表示します。
削除に失敗した場合はエラー内容をアラートで表示します。

```
// お気に入り登録ではオブジェクトデータと userAccessToken が必須項目です
var objectData = {
    object_id: "object1",
    code: "0001"
};
var userAccessToken = "userAccessTokenXXXX";

// お気に入りを削除する（RKZClient 変数は初期化済みとする）
RKZClient.deleteFavoriteToObjectData(objectData, userAccessToken,
    function(statusCode) {
        // 成功時にアラートでメッセージを表示します
        alert("お気に入りを削除しました。");

    }, 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, "    "));
    }
);
```

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 "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。

```
// デフォルトタイムアウト時間の設定
RKZClient.setDefaultTimeout(10,
    function(locale) {
        alert("SUCCESS");
    }, function(error) {
        // 10秒以上かかる場合はエラーとなる
        alert(JSON.stringify(error));
    });
```

更新履歴

版数	日付	更新内容
第 1 版	2017/01/27	◆ Ver1.0.0 対応版　初版。
第 2 版	2017/02/01	◆ Ver1.0.1 対応版 ◆ Monaca でのプラグインインストール方法変更
第 3 版	2019/06/06	◆ Ver1.2.0 対応版