SDK リファレンスマニュアル

iOS 版

Ver2.3.0

第 4 版

PEOPLE SOFTWARE

内容

Ver 2.3.0 で WKWebView に対応しました。

UIWebView を利用している場合、以下の日程で App Store でアプリ申請を受け付けてくれなくなります。

- 新規リリース　→**2020 年 04 月**
- アップデート　→**2020 年 12 月**

参考サイト：

- https://developer.apple.com/news/?id=12232019b
- 2020 年 4 月以降の iOS アプリ申請について
  https://qiita.com/keeey/items/d591aa212992b3281458#wkwebview-%E5%AF%BE%E5%BF%9C

Ver2.3.0 にて、UIWebView の利用を廃止し、WKWebView に対応しました。
BaaS@rakuza にて、iOS アプリケーションを作成する場合は、Ver2.3.0 を利用して下さい。
また、既に公開されているアプリケーションの場合は、Ver2.3.0 への差し替えをお願いします。

クイックスタート

BaaS@kuraza SDK for iOS の利用方法

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

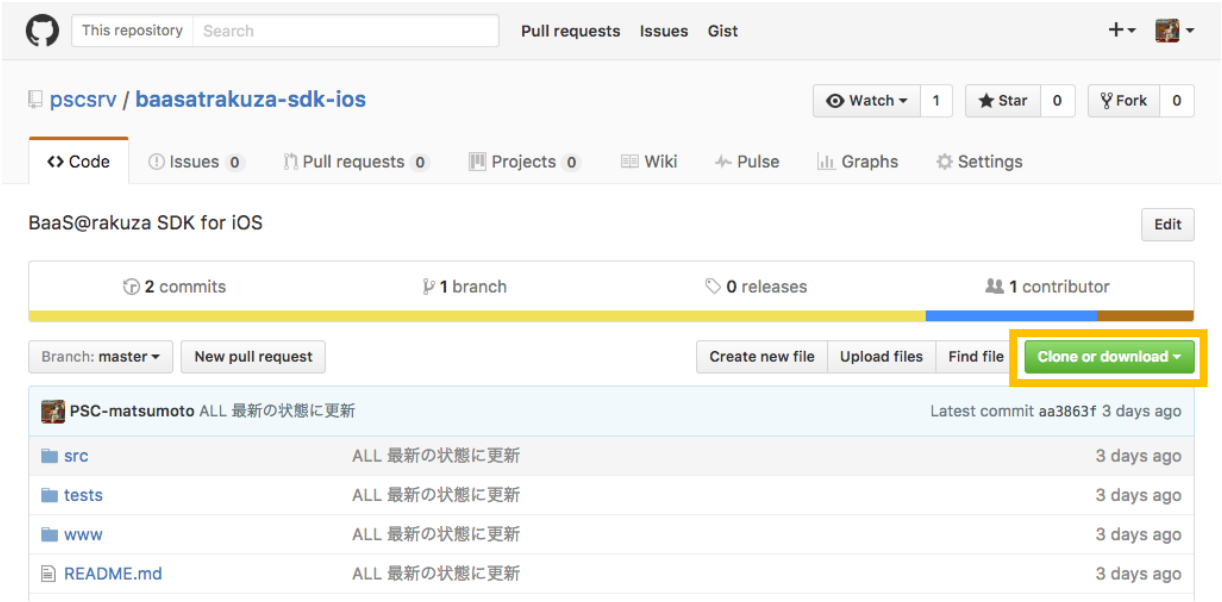
手順書作成環境

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。

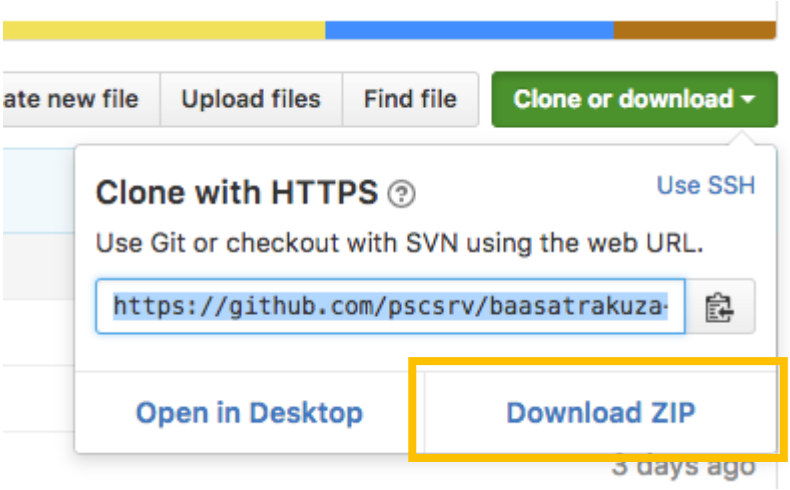
- Xcode Version 6.3.2
- OS X Yosemite Version 10.10.3

BaaS@rakuza SDK for iOS 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-ios 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。

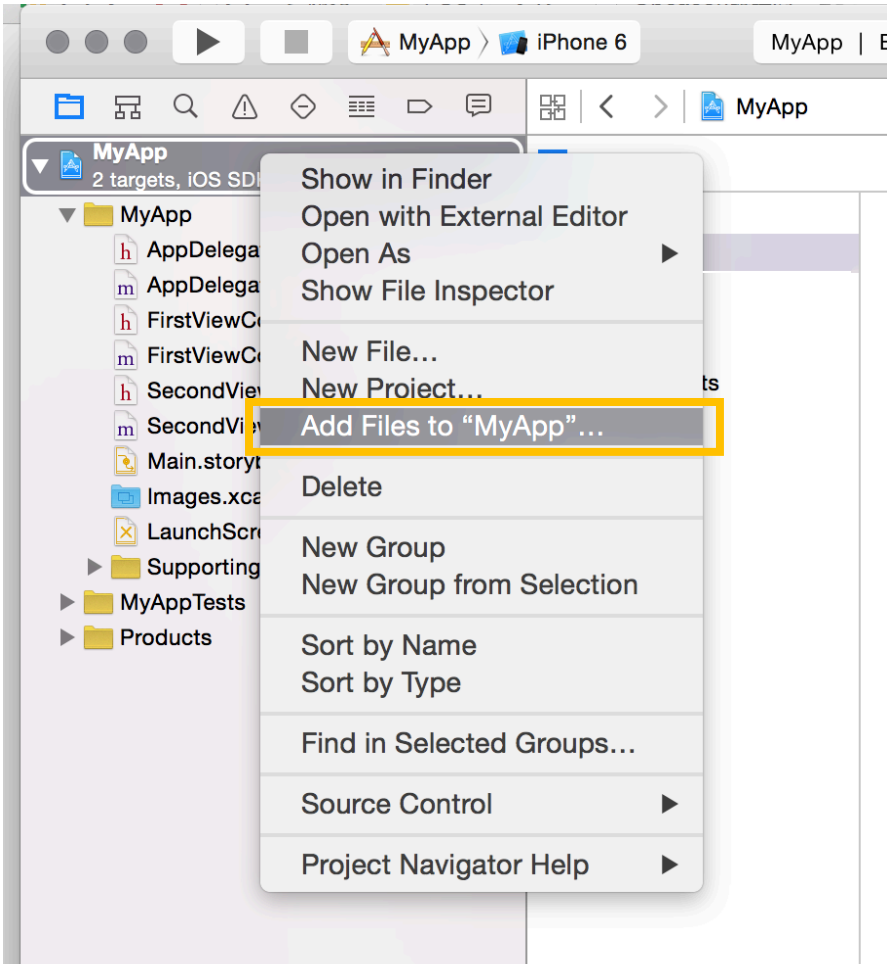
提供ファイルの構成は以下になります。

```
baasatrakuza-sdk-ios
  ├docs
  |   ├appledoc.zip
  |   └BaaSAtRakuzaSDK リファレンスマニュアル_iOS_x.pdf
  └libs
      ├libBaaSAtRakuza.a
      └include
```

プロジェクトへの設定

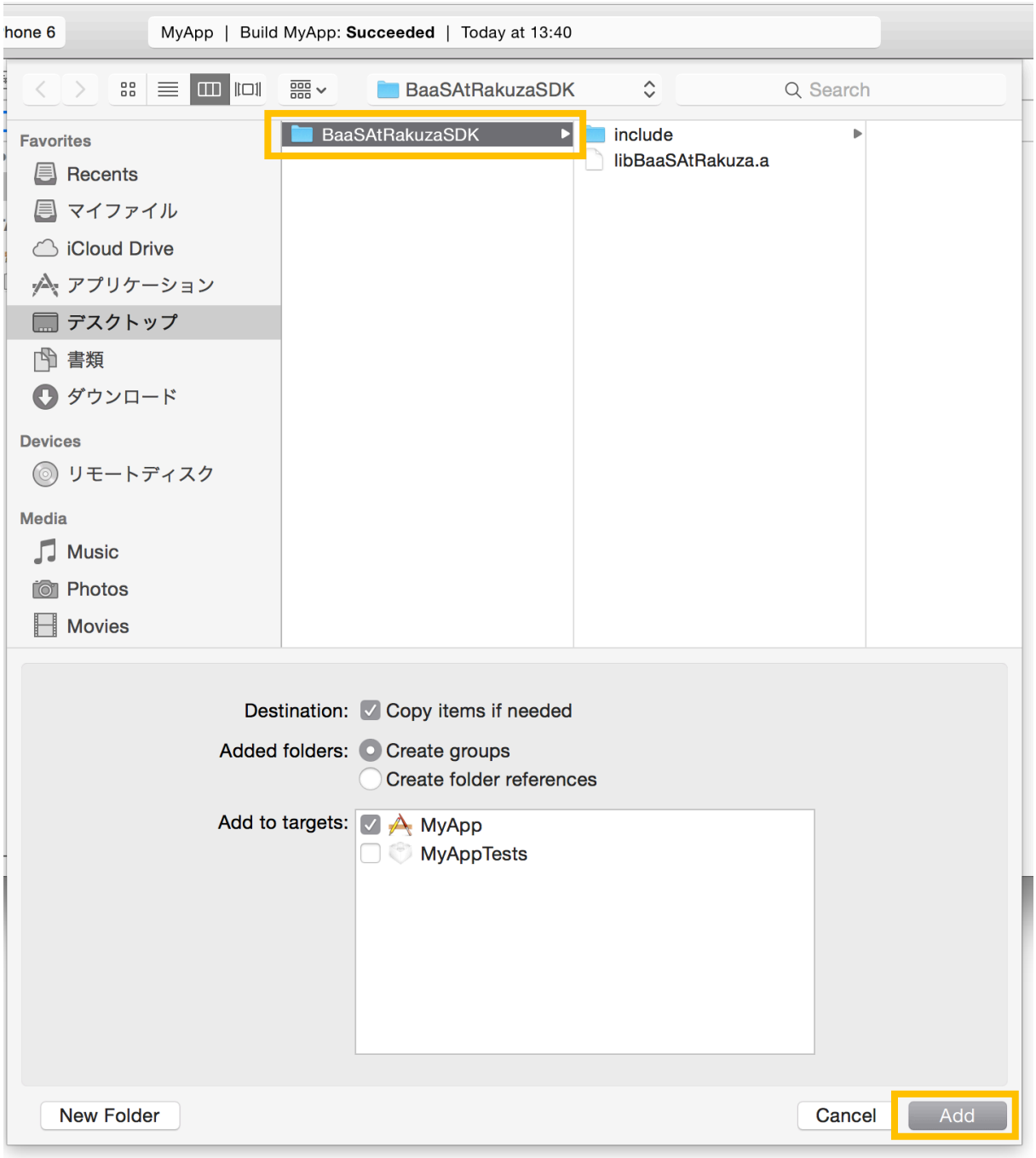
BaaS@rakuza SDK をプロジェクトに追加する

左ペインのプロジェクトを右クリックし "Add Files to "[プロジェクト名] "... " をクリックします。



追加するファイルを選択する画面に遷移しますので、任意のディレクトリに展開した BaaS@rakuza SDK の libs 以下を選択して "Add" をクリックします。

※注意点 Added folders: "Create Groups" が選択されていることをご確認ください。
プロジェクトにライブラリファイルを含める場合、Destination: "Copy items if needed" にチェックをしてください。

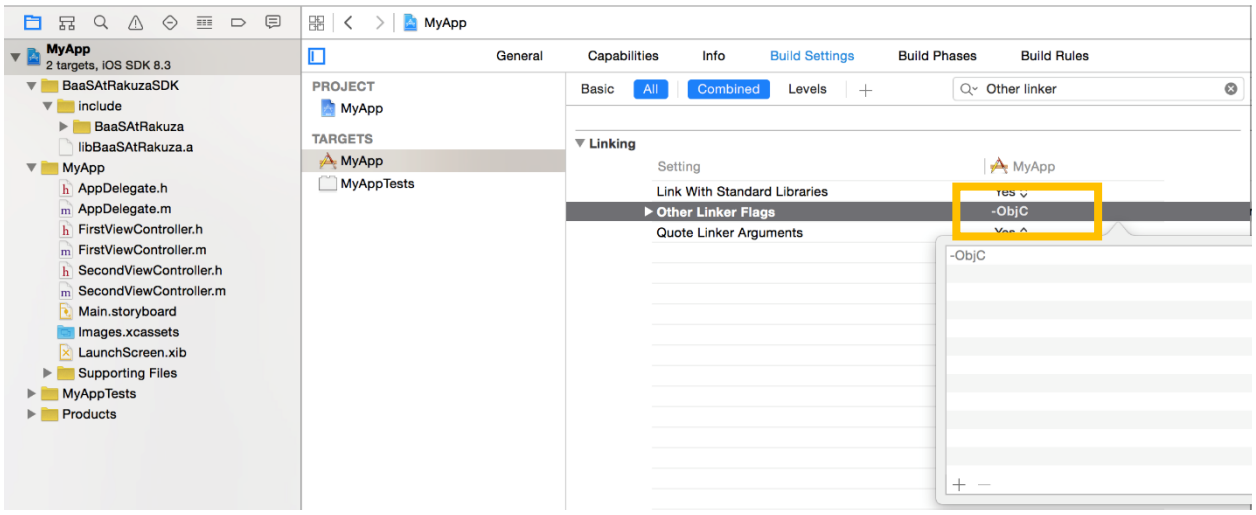


Other Linker Flags の設定

左ペインのプロジェクトを選択し、TARGETS -> Build Settings Other Linker Flags に以下の設定を追加します。

-ObjC

※注意点 静的ライブラリで Objective-c の"カテゴリ"を使用しているため、必ず必要となります。

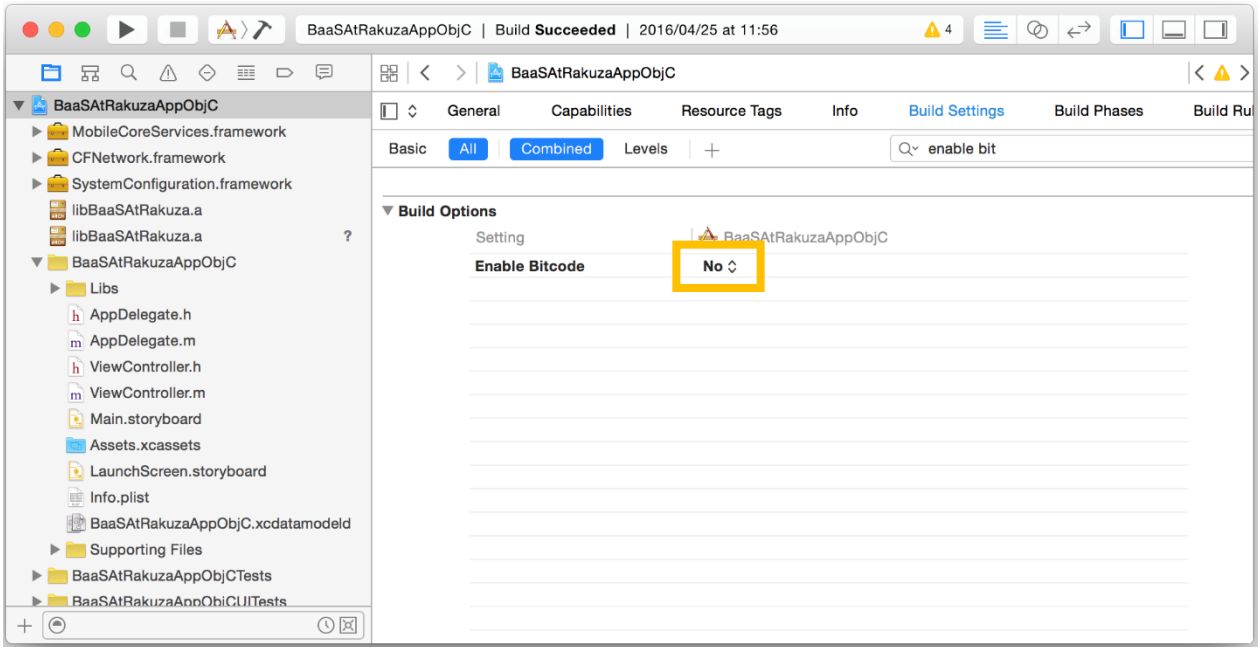


Enable Bitcode の設定

Xcode 7 以降を利用している場合に必要となる設定です。
以下のエラーが検出された場合、プロジェクトの設定を変更する必要があります。

ld: 'ライブラリパス/ライブラリ名' does not contain bitcode. You must rebuild it with bitcode enabled
(Xcode setting ENABLE_BITCODE), obtain an updated library from the vendor, or disable bitcode for this target. for architecture arm64
clang: error: linker command failed with exit code 1 (use -v to see invocation)

プロジェクトを選択し、TARGETS -> Build Settings > Enable Bitcode に"No"を設定します。



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 SDK for iOS を利用する際には、RKZService 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行うことで、BaaS@rakuza の API を利用すること出来るようになります。

ライブラリを読み込む

Appdelegate.m の冒頭に以下のコードを記載してください。

```
#import "RKZService.h"
```

RKZService を初期化する

ここでは、BaaS@rakuza SDK for iOS を使用するうえで重要な RKZService の初期化を説明します。
BaaS@rakuza SDK for iOS では RKZService の初期化は Appdelegate.m の application:didFinishLaunchingWithOptions: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。

Appdelegate.m を開き、application:didFinishLaunchingWithOptions:メソッドに以下のコードを追加します。

```
RKZResponseStatus *responseStatus = [[RKZService sharedInstance]setTenantKey:@"配布したテナントキー"];
if (responseStatus.isSuccess) {
  // RKZService 初期化成功です
  // 任意の正常処理を行ってください
} else {
  // RKZService 初期化失敗です
  // 任意のエラー処理を行ってください
}
```

以上で BaaS@rakuza SDK for iOS の API を利用する準備が完了しました。

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
RKZService の getDataList:searchConditionArray:sortConditionArray:withBlock:
で行います。

お気に入り情報登録（addFavoriteToNews）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ログ出力します。
取得に失敗した場合はエラー内容をログ出力します。

```
NSString *objectId = @"object1";   // オブジェクト ID を指定します
// name 項目に対して前方一致条件を指定 ※ name LIKE "サンプル*"
RKZSearchCondition *searchCondition1 = [RKZSearchCondition initWithSearchConditionType:RKZSearchConditionLikeB
efore searchColumn:@"name" searchValueArray:[@[@"サンプル"]mutableCopy]];

// short_name 項目に対して等価条件を指定 ※ short_name = "サンプル"
RKZSearchCondition *searchCondition2 = [RKZSearchCondition initWithSearchConditionType:RKZSearchConditionEqual
                                    searchColumn:@"short_name"
                                searchValueArray:[@[@"サンプル"]mutableCopy]];
// 複数条件を指定
NSMutableArray *searchConditions = [@[searchCondition1, searchCondition2]mutableCopy];

// sort_no 項目の昇順を指定
RKZSortCondition *sortCondition1 = [RKZSortCondition initWithSortType:RKZSortTypeAsc sortColumn:@"sort_no"];
```

```
NSMutableArray *sortConditions = [@[sortCondition1]mutableCopy];

// データオブジェクト データ取得
[[RKZService sharedInstance]getDataList:objectId
                  searchConditionArray:searchConditions
                  sortConditionArray:sortConditions
                  withBlock:^(NSMutableArray *rkzObjectDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    for (RKZObjectData *rkzObjectData in rkzObjectDataArray) {
      // 成功時には userData に登録内容が格納されて返却されます
      NSLog(@"rkzObjectData.code          :%@", rkzObjectData.code);
      NSLog(@"rkzObjectData.name          :%@", rkzObjectData.name);
      NSLog(@"rkzObjectData.short_name      :%@", rkzObjectData.short_name);
      NSLog(@"rkzObjectData.sort_no         :%@", rkzObjectData.sort_no);
      NSLog(@"rkzObjectData.attributes['company'] :%@", rkzObjectData.attributes[@"company"]);
      NSLog(@"rkzObjectData.attributes['hoge']  :%@", rkzObjectData.attributes[@"hoge"]);
    }
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"message    :%@", responseStatus.message);
  }
}];
```

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
RKZSearchConditionIn	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
MyFavoriteOnlyForFavorite	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
NotMyFavoriteForFavorite	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
AllForFavorite	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
AlreadyReadForReadedNews	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
NonreadForReadedNews	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
AllForReadedNews	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
RKZSortTypeAsc	昇順
RKZSortTypeDesc	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 RKZService の getData:code:withBlock で行います。データは RKZObjectData として返却されます。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *objectId = @"object01"; // オブジェクト ID を指定します
NSString *code = @"0001";         // コードを指定します

// データオブジェクト データ取得
[[RKZService sharedInstance]getData:objectId code:code
                      withBlock:^(RKZObjectData *rkzObjectData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    // 成功時には userData に登録内容が格納されて返却されます
    NSLog(@"rkzObjectData.code          :%@", rkzObjectData.code);
    NSLog(@"rkzObjectData.name          :%@", rkzObjectData.name);
```

```
    NSLog(@"rkzObjectData.short_name      :%@", rkzObjectData.short_name);
    NSLog(@"rkzObjectData.sort_no         :%@", rkzObjectData.sort_no);
    NSLog(@"rkzObjectData.attributes['company'] :%@", rkzObjectData.attributes[@"company"]);
    NSLog(@"rkzObjectData.attributes['hoge']  :%@", rkzObjectData.attributes[@"hoge"]);
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

オブジェクトデータを登録する

オブジェクトのデータ登録は RKZService の addData:withBlock: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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 登録する情報作成
RKZObjectData *data = [[RKZObjectData alloc]init];
data.object_id = @"object01";
data.name = @"名称";
data.attributes = [@{@"company" : @"People Software Corp.",
                @"hoge"      : @"hoge"
                }mutableCopy];
// データオブジェクト データ登録
[[RKZService sharedInstance]addData:data
                    withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ登録成功
  } else {
    // データオブジェクト データ登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

オブジェクトデータを編集する

オブジェクトのデータ編集は RKZService の editData:withBlock 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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 編集する情報作成 ※実際の実装では getData, getDataList の取得結果を利用してください
RKZObjectData *data = [[RKZObjectData alloc]init];
data.object_id = @"object01";
data.code = @"0001";
data.name = @"名称";
data.attributes = [@{@"company"  : @"People Software Corp.",
                     @"hoge"     : @"hoge"
                   }mutableCopy];
// データオブジェクト データ編集
[[RKZService sharedInstance]editData:data
                     withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ編集成功
  } else {
    // データオブジェクト データ編集失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の deleteData:searchConditions:withBlock 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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *objectId = @"coupon";     // テーブル名を指定します
NSString *code = @"0010";     // コードを指定します
NSMutableArray *searchConditions = nil;      // 削除条件があれば指定します。Nil を指定すると全件削除します。

// データオブジェクト データ削除
[[RKZService sharedInstance]deleteData:objectId
                  searchConditions:searchConditions
                        withBlock:^(NSNumber deleteCount, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ削除成功
  } else {
    // データオブジェクト データ削除失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService.getDataListWithRelationObjects:searchConditionArray:sortConditionArray:withBlock か、RKZService.getDataListWithRelationObjects:treeCount:serarchConditionArray:sortConditionArray:withBlock 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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *objectId = @"object01"; // オブジェクト ID を指定します
NSNumber *treeCount = 2;          // 2 階層分取得します

// データオブジェクト データ取得
[[RKZService sharedInstance]getDataListWithRelationObjects:objectId treeCount:treeCount
```



```
                    searchConditionArray:nil sortConditionArray:nil
                    withBlock:^(NSMutableArray *rkzObjectDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    for (RKZObjectData *rkzObjectData in rkzObjectDataArray) {
      // 成功時には userData に登録内容が格納されて返却されます
      NSLog(@"rkzObjectData.code          :%@", rkzObjectData.code);
      NSLog(@"rkzObjectData.name          :%@", rkzObjectData.name);
      NSLog(@"rkzObjectData.short_name      :%@", rkzObjectData.short_name);
      NSLog(@"rkzObjectData.sort_no         :%@", rkzObjectData.sort_no);
      NSLog(@"rkzObjectData.attributes['company'] :%@", rkzObjectData.attributes[@"company"]);
      NSLog(@"rkzObjectData.attributes['hoge']  :%@", rkzObjectData.attributes[@"hoge"]);
    }
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、
RKZService.getPaginateDataList:limit:offset:searchConditions:sortConditions:withBlock 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（addFavoriteToNews）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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *objectId = @"object01";  // オブジェクト ID を指定します
NSNumber *limit = 10;              // 取得するデータの件数を指定します
NSNumber *offset = 0;              // データの取得位置を指定します

// データオブジェクト データ取得
[[RKZService sharedInstance]getPaginateDataList:objectId limit:limit offset:offset
                    searchConditions:nil sortConditions:nil
                    extensionAttribute:nil
                    withBlock:^( RKZPagingData *rkzPagingData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    // 成功時には userData に登録内容が格納されて返却されます
    NSLog(@"rkzPagingData.limit          :%@", rkzPagingData.limit);
    NSLog(@"rkzPagingData.offset         :%@", rkzPagingData.offset);
    NSLog(@"rkzPagingData.result_cnt     :%@", rkzPagingData.result_cnt);
    NSLog(@"rkzPagingData.datas          :%@", rkzPagingData.datas);
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、
RKZService.getDataWithLocation:code:location:spotFieldName:withBlock(1 レコード取得)か、
RKZService.getDataListWithLocation:location:spotFieldName:searchConditionArray:sortConditionArray:withBlock(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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *objectId = @"object01"; // オブジェクト ID を指定します
NSString *code = @"0001";         // コードを指定します
RKZLocation *location = [[RKZLocation alloc] init];     // 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.latitude = 34.600917;     // 緯度を指定
location.longitude = 133.765784;   // 経度を指定

// データオブジェクト データ取得
[[RKZService sharedInstance]getDataWithLocation:objectId code:code location:location spotFieldName:nil
                    withBlock:^(RKZObjectData *rkzObjectData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    // 成功時には userData に登録内容が格納されて返却されます
    NSLog(@"rkzObjectData.code          :%@", rkzObjectData.code);
    NSLog(@"rkzObjectData.name          :%@", rkzObjectData.name);
    NSLog(@"rkzObjectData.short_name      :%@", rkzObjectData.short_name);
    NSLog(@"rkzObjectData.sort_no         :%@", rkzObjectData.sort_no);
    NSLog(@"rkzObjectData.attributes['company'] :%@", rkzObjectData.attributes[@"company"]);
    NSLog(@"rkzObjectData.attributes['hoge']  :%@", rkzObjectData.attributes[@"hoge"]);
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、
RKZService.getFieldDataList:visibleFieldOnly:withBlock で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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

// データオブジェクト データ取得
[[RKZService sharedInstance]getFieldDataList:objectId visibleFieldOnly:visibleOnly
                    withBlock:^(NSMutableArray * rkzFieldDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // データオブジェクト データ取得成功
    for (RKZFieldData *rkzFieldData in rkzFieldDataArray) {
      // 成功時には userData に登録内容が格納されて返却されます
      NSLog(@"rkzFieldData.field_name         :%@", rkzFieldData.field_name);
      NSLog(@"rkzFieldData.label_str          :%@", rkzFieldData.label_str);
    }
  } else {
    // データオブジェクト データ取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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。

```
NSString *objectId = @"object1";   // オブジェクト ID を指定します
NSMutableArray *searchConditions = nil;
NSMutableArray *sortConditions = nil;

// extensionAttribute 引数を設定
RKZObjectDataExtensionAttribute *extensionAttribute = [RKZObjectDataExtensionAttribute new];
extensionAttribute.user_access_token = @"USER_ACCESS_TOKEN";
extensionAttribute.show_favorite = YES;   // お気に入り情報を復帰する
extensionAttribute.show_favorite_summary = YES;   // お気に入り件数を復帰する

// データオブジェクト データ取得
[[RKZService sharedInstance]getDataList:objectId
                    searchConditionArray:searchConditions
                      sortConditionArray:sortConditions
                      extensionAttribute:extensionAttribute
                               withBlock:^(NSMutableArray *rkzObjectDataArray, RKZResponseStatus *responseStatu
s) {
                                   if (responseStatus.isSuccess) {
                                       // データオブジェクト データ取得成功
                                       for (RKZObjectData *rkzObjectData in rkzObjectDataArray) {
                                           NSDictionary *sysFavorite = [rkzObjectData.attributes objectForKey:@
"sys_favorite"];
                                           // 成功時には userData に登録内容が格納されて返却されます
                                           NSLog(@"rkzObjectData.is_favorite :%@", [sysFavorite objectForKey:@"
is_favorite"]);
                                           NSLog(@"rkzObjectData.favorite_date :%@", [sysFavorite objectForKey:
@"favorite_date"]);
                                       }
                                   } else {
```

```
                                // データオブジェクト データ取得失敗
                                // 失敗時には responseStatus にエラー情報が格納されて返却されます
                                NSLog(@"statusCode :%@", @(responseStatus.statusCode));
                                NSLog(@"message :%@", responseStatus.message);
                            }
                        }];
```

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 RKZService の registUser で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
RKZUserData *userData = [[RKZUserData alloc] init];
userData.user_name = @"ピープル太郎";  // 名称を指定します
userData.nick_name = @"ピープル君";  // ニックネームを指定します
userData.mail_address_1 = @"hogehoge@pscsrv.co.jp";  // メールアドレスを指定します
userData.attributes = [@{   // 自由項目を指定します
            @"company"  : @"People Software Corp.",
            @"hoge"   : @"hoge"
            }mutableCopy];
// ユーザー登録 API の実行
[[RKZService sharedInstance]registUser:userData
                    withBlock:^(RKZUserData *userData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // ユーザー登録成功
    // 成功時には userData に登録内容が格納されて返却されます
    NSLog(@"userData.user_name         :%@", userData.user_name);
    NSLog(@"userData.nick_name          :%@", userData.nick_name);
    NSLog(@"userData.mail_address_1       :%@", userData.mail_address_1);
    NSLog(@"userData.user_access_token     :%@", userData.user_access_token);
    NSLog(@"userData.attributes['company'] :%@", userData.attributes[@"company"]);
```

```
    NSLog(@"userData.attributes['hoge']    :%@", userData.attributes[@"hoge"]);
  } else {
    // ユーザー登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

ユーザー情報を取得する

ユーザー情報取得は RKZService の getUser で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// ユーザー情報を取得
[[RKZService sharedInstance]getUser:userAccessToken withBlock:^(RKZUserData *userData, RKZResponseStatus *resp
onseStatus) {
  if (responseStatus.isSuccess) {
    // ユーザー情報取得成功
    // 成功時には userData に登録内容が格納されて返却されます
    NSLog(@"userData.user_name       :%@", userData.user_name);
    NSLog(@"userData.nick_name       :%@", userData.nick_name);
    NSLog(@"userData.mail_address_1    :%@", userData.mail_address_1);
    NSLog(@"userData.attributes['company'] :%@", userData.attributes[@"company"]);
    NSLog(@"userData.attributes['hoge']  :%@", userData.attributes[@"hoge"]);
  } else {
    // ユーザー情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の editUser:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
RKZUserData *userData = [[RKZUserData alloc] init];
userData.user_access_token = @"userAccessTokenXXXX";    // ユーザアクセストークンを指定します
userData.user_name = @"バース太郎";    // 氏名の変更

[[RKZService sharedInstance] editUser:userData withBlock: ^(RKZUserData *userData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // ユーザー情報変更成功
    // 成功時には userData に最新のユーザー情報が格納されて返却されます
    NSLog(@"userData.user_name       :%@", userData.user_name);
    NSLog(@"userData.nick_name       :%@", userData.nick_name);
    NSLog(@"userData.mail_address_1    :%@", userData.mail_address_1);
    NSLog(@"userData.attributes['company'] :%@", userData.attributes[@"company"]);
    NSLog(@"userData.attributes['hoge']  :%@", userData.attributes[@"hoge"]);
  } else {
    // ユーザー情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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
RKZService の registModelChangeCode:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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。
NSString *userAccessToken = @"userAccessTokenXXXX";

[[RKZService sharedInstance] registModelChangeCode:userAccessToken withBlock: ^(NSString *modelChangeCode, NSDate *limitDate, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // 機種変更認証コード生成成功
    // 成功時には modelChangeCode に発行された認証コードが格納されて返却されます。
    NSLog(@"modelChangeCode        :%@", modelChangeCode);
    // limitDate には、認証コードの有効期限が格納されて返却されます。
    NSLog(@"limitDate        :%@", limitDate);
  } else {
    // ユーザー情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の registModelChangeCode:password:limitCode:limitMinute:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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。
NSString *userAccessToken = @"userAccessTokenXXXX";

// 任意項目(パスワード、桁数、有効時間)の設定を行います。
NSString *password = @"ユーザーしかしらないぱすわーど";
NSNumber *limitCode = 8;            // 認証コードの桁数は 8 桁
NSNumber limitMinute = 60 * 24;     // 認証コードの有効時間は 1 日(単位：分)

[[RKZService sharedInstance] registModelChangeCode:userAccessToken password:password limitCode:limitCode limitMinute:limitMinute withBlock: ^(NSString *modelChangeCode, NSDate *limitDate, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // 機種変更認証コード生成成功
```

```
    // 成功時には modelChangeCode に発行された認証コードが格納されて返却されます。
    NSLog(@"modelChangeCode        :%@", modelChangeCode);
    // limitDate には、認証コードの有効期限が格納されて返却されます。
    NSLog(@"limitDate        :%@", limitDate);
  } else {
    // ユーザー情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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RKZServiceのauthModelChangeCod:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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
NSString *modelChangeCode = @"認証コード";

[[RKZService sharedInstance] authModelChangeCode:modelChangeCode withBlock: ^(RKZUserData *userData, RKZRespon
seStatus *responseStatus){
  if (responseStatus.isSuccess) {
    // 機種変更認証成功
    // 成功時には userData にユーザー情報が格納されて返却されます。
    NSLog(@"userData.user_name        :%@", userData.user_name);
  } else {
    // 機種変更認証失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の
authModelChangeCode:password:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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
NSString *modelChangeCode = @"認証コード";

// 任意項目(パスワード)の設定を行います。
NSString *password = @"ユーザーしかしらないぱすわーど";

[[RKZService sharedInstance] authModelChangeCode:modelChangeCode password:password withBlock: ^(RKZUserData *u
serData, RKZResponseStatus *responseStatus){
  if (responseStatus.isSuccess) {
    // 機種変更認証成功
    // 成功時には userData にユーザー情報が格納されて返却されます。
    NSLog(@"userData.user_name        :%@", userData.user_name);
  } else {
    // 機種変更認証失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の updateUserAccessToken:withBlock: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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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

[[RKZService sharedInstance] updateUserAccessToken:userAccessToken withBlock: ^(NSString *newUserAccessToken,
RKZResponseStatus *responseStatus){
  if (responseStatus.isSuccess) {
    // ユーザーアクセストークン更新成功
    // 成功時には newUserAccessToken に新しいユーザーアクセストークンが格納されて返却されます。
    NSLog(@"newUserAccessToken :%@", newUserAccessToken);
  } else {
    // ユーザーアクセストークン更新失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の beginUpdateUserAccessToken:withBlock:で新しいユーザーアクセストークンを仮発行して、commitUpdateUserAccessToken:withBlock: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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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

[[RKZService sharedInstance] beginUpdateUserAccessToken:userAccessToken withBlock: ^(NSString *newUserAccessToken, RKZResponseStatus *responseStatus){
  if (responseStatus.isSuccess) {
    // ユーザーアクセストークン仮発行成功
    // 成功時には newUserAccessToken に新しいユーザーアクセストークンが格納されて返却されます。
    [[RKZService sharedInstance] commitUpdateUserAccessToken:userAccessToken withBlock: ^(NSString *newUserAccessToken, RKZResponseStatus *responseStatus){
      if (responseStatus.isSuccess) {
        // ユーザーアクセストークン確定成功
        // 成功時には newUserAccessToken に新しいユーザーアクセストークンが格納されて返却されます。
        NSLog(@"newUserAccessToken :%@", newUserAccessToken);
      } else {
        // 失敗時には responseStatus にエラー情報が格納されて返却されます
        NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
        NSLog(@"message     :%@", responseStatus.message);
        NSLog(@"detailMessage:%@", responseStatus.detailMessage);
      }
    }];
  } else {
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の getContactList:searchConditionArray:sortConditionArray:withBlock で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。

```
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";

// コンタクト取得
// 検索条件、ソート条件には nil を指定しています。
[[RKZService sharedInstance] getContactList:userAccessToken searchConditionArray:nil sortConditionArray:nil wi
thBlock:^(NSMutableArray *contactDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // コンタクト取得成功
    for (RKZContactData *contactData in contactDataArray) {
      NSLog(@"contact_no :%@", contactData.contact_no);
      NSLog(@"contact_date :%@", contactData.contact_date);
      NSLog(@"contact_class_cd :%@", contactData.contact_class_cd);
      NSLog(@"contact_class_name :%@", contactData.contact_class_name);
      NSLog(@"contact_method_class_cd :%@", contactData.contact_method_class_cd);
      NSLog(@"contact_method_class_name :%@", contactData.contact_method_class_name);
      NSLog(@"contact_item_no :%@", contactData.contact_item_no);
      NSLog(@"contact_item_name :%@", contactData.contact_item_name);
```

```
        NSLog(@"entry_no :%@", contactData.entry_no);
        NSLog(@"status_cd :%@", contactData.status_cd);
        NSLog(@"status_name :%@", contactData.status_name);
        NSLog(@"place_cd :%@", contactData.place_cd);
        NSLog(@"point :%@", contactData.point);
        NSLog(@"remarks :%@", contactData.remarks);
        NSLog(@"deposit_no :%@", contactData.deposit_no);
        NSLog(@"beacon_id :%@", contactData.beacon_id);
        NSLog(@"beacon_spot_cd :%@", contactData.beacon_spot_cd);
        NSLog(@"beacon_spot_name :%@", contactData.beacon_spot_name);
        NSLog(@"rssi :%@", contactData.rssi);
        NSLog(@"coupon_cd :%@", contactData.coupon_cd);
        NSLog(@"quantity :%@", contactData.quantity);
        NSLog(@"stamp_rally_cd :%@", contactData.stamp_rally_cd);
        NSLog(@"stamp_rally_name :%@", contactData.stamp_rally_name);
        NSLog(@"stamp_rally_spot_cd :%@", contactData.stamp_rally_spot_cd);
        NSLog(@"stamp_rally_spot_name :%@", contactData.stamp_rally_spot_name);
        NSLog(@"attributes['hoge']  :%@", contactData.attributes[@"hoge"]);
    }
  } else {
    // コンタクト履歴取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の addContact:contactData:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// コンタクト履歴を作成
RKZContactData *contactData = [RKZContactData new];
contactData.contact_class_cd = @"0005";
contactData.contact_method_class_cd = @"0009";
contactData.beacon_id = @"DB000001";
```

```
contactData.attributes[@"testfield"] = @"test";

// コンタクト履歴登録
[[RKZService sharedInstance] addContact:userAccessToken contactData:contactData withBlock:^(RKZApiStatusCode s
tatusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // コンタクト履歴登録成功
  } else {
    // コンタクト履歴登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
getNewsList:searchConditionArray:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
拡張属性は null を指定しています。

```
// 取得件数の上限を指定します
NSNumber *limit = @10;
// お知らせ情報取得
[[RKZService sharedInstance]getNewsList:limit
                 searchConditionArray:nil
                   sortConditionArray:nil
                   extensionAtttribute:nil
                        withBlock:^(NSMutableArray *newsDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    //お知らせ情報取得成功
    for (RKZNewsData *newsData in newsDataArray) {
      NSLog(@"news_id:%@", newsData.news_id);
    }
  } else {
    // お知らせ情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
```

```
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getReleasedNewsList:searchConditionArray:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
拡張属性は null を指定しています。

```
// 取得件数の上限を指定します
NSNumber *limit = @10;
// お知らせ情報取得
[[RKZService sharedInstance]getReleasedNewsList:limit
                    searchConditionArray:nil
                      sortConditionArray:nil
                      extensionAtttribute:nil
                             withBlock:^(NSMutableArray *newsDataArray,
                                     RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    //お知らせ情報取得成功
    for (RKZNewsData *newsData in newsDataArray) {
      NSLog(@"news_id:%@", newsData.news_id);
    }
  } else {
    // お知らせ情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
```

```
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getNews:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// お知らせ ID を指定します
NSString *newsId = @"1";
// お知らせ情報取得
[[RKZService sharedInstance]getNews:newsId
                        withBlock:^(RKZNewsData *newsData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // お知らせ情報取得成功
    NSLog(@"news_id:%@", newsData.news_id);
  } else {
    // お知らせ情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
getNewsReadHistory:userAccessToken:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// お知らせ ID を指定します
NSString *newsId = @"1";
// ユーザーアクセストークンを指定します
```

```
NSString *userAccessToken = @"userAccessTokenXXXX";
// お知らせ既読情報取得
[[RKZService sharedInstance]getNewsReadHistory:newsId
                                userAccessToken:userAccessToken
                                     withBlock:^(RKZNewsReadHistoryData *newsReadHistoryData,
                                                 RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // お知らせ既読情報取得成功
    NSLog(@"news_id:%@", newsReadHistoryData.news_id);
  } else {
    // お知らせ既読情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
getNewsReadHistoryList:withBlock で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// お知らせ既読情報取得
[[RKZService sharedInstance]getNewsReadHistoryList:userAccessToken
                                         withBlock:^(NSMutableArray *newsReadHistoryDataArray, RKZResponseStatus *responseStatu
s) {
  if (responseStatus.isSuccess) {
    // お知らせ既読情報取得成功
    for (RKZNewsReadHistoryData *newsReadHistoryData in newsReadHistoryDataArray) {
      NSLog(@"news_id:%@", newsReadHistoryData.news_id);
    }
  } else {
    // お知らせ既読情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
```

```
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getSegmentNewsList:userAccessToken:onlyMatchSegment:searchConditionArray:sortConditionArray:withBlock で行います。

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。
拡張属性は null を指定しています。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
NSString *userAccessToken = @"userAccessTokenXXXX";    // ユーザーアクセストークンを指定します
NSNumber *limit = 10;      // 取得するお知らせの件数を指定します
BOOL *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

[[RKZService sharedInstance]getSegmentNewsList:limit userAccessToken:userAccessToken
                    onlyMatchSegment:onlyMatch
                    extensionAtttribute:nil
                    withBlock:^( NSMutableArray *newsDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // お知らせ既読情報取得成功
    for (RKZNewsData *newsData in newsDataArray) {
      NSLog(@"news_id:%@", newsData.news_id);
    }
  } else {
    // お知らせ既読情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
registNewsReadHistory:userAccessToken:readData:withBlock: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。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// お知らせ ID を指定します
NSString *newsId = @"1";
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// 既読日時を指定します
NSDate *readDate = [NSDate date];
// お知らせ既読情報登録
[[RKZService sharedInstance]registNewsReadHistory:newsId
                          userAccessToken:userAccessToken
                                readDate:readDate
                                withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // お知らせ既読情報取得成功
  } else {
    // お知らせ既読情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
readNews:userAccessToken:withBlock: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。

```
RKZNewsExtensionAttribute *extensionAttribute = [RKZNewsExtensionAttribute new];
extensionAttribute.user_access_token = @"userAccessTokenXXXX";
```

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// お知らせ ID を指定します
NSString *newsId = @"1";
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// お知らせ既読情報登録
[[RKZService sharedInstance]readNews:newsId
                          userAccessToken:userAccessToken
                          withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // お知らせ既読情報取得成功
  } else {
    // お知らせ既読情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の registPushDeviceToken:deviceToken:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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

[[RKZService sharedInstance] registPushDeviceToken:userAccessToken withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // プッシュデバイストークン登録成功
  } else {
    // プッシュデバイストークン登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、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設定は RKZService の
clearPushDeviceToken:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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

[[RKZService sharedInstance] clearPushDeviceToken:userAccessToken withBlock:^(RKZApiStatusCode statusCode, RKZ
ResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // プッシュデバイストークン削除成功
  } else {
    // プッシュデバイストークン削除失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getBeaconList:sortConditionArray:withBlock: で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
[[RKZService sharedInstance] getBeaconList:nil sortConditionArray:nil
                          withBlock:^(NSMutableArray *beaconDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // ビーコン取得成功
    for (RKZBeaconData *beaconData in beaconDataArray) {
      NSLog(@"code     :%@", beaconData.code);
      NSLog(@"name     :%@", beaconData.name);
      NSLog(@"short_name    :%@", beaconData.short_name);
      NSLog(@"beacon_id :%@", beaconData.beacon_id);
      NSLog(@"beacon_type_cd   :%@", beaconData.beacon_type_cd);
      NSLog(@"major   :%@", beaconData.major);
      NSLog(@"minor   :%@", beaconData.minor);
      NSLog(@"beaconData.attributes['hoge']   :%@", beaconData.attributes[@"hoge"]);
    }
  } else {
    // ビーコン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
```

```
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getSpotList:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
[[RKZService sharedInstance] getSpotList:nil sortConditionArray:nil withBlock: ^(NSMutableArray *spotDataArra
y, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // ビーコン取得成功
    for (RKZSpotData *spotData in spotDataArray) {
      NSLog(@"code     :%@", spotData.code);
      NSLog(@"name     :%@", spotData.name);
    }
  } else {
    // ビーコン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```


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、RKZService の getCouponList:sortConditionArray:withBlock: で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
[[RKZService sharedInstance] getCouponList:nil sortConditionArray:nil
                          withBlock:^(NSMutableArray *couponDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // クーポン取得成功
    for (RKZCouponData *couponData in couponDataArray) {
      NSLog(@"code     :%@", couponData.code);
      NSLog(@"name     :%@", couponData.name);
      NSLog(@"image    :%@", couponData.image);
      NSLog(@"image_url :%@", couponData.image_url);
      NSLog(@"possible_from_dte:%@", couponData.possible_from_dte);
      NSLog(@"possible_to_dte  :%@", couponData.possible_to_dte);
      NSLog(@"enable_from_dte  :%@", couponData.enable_from_dte);
      NSLog(@"enable_to_dte  :%@", couponData.enable_to_dte);
    }
  } else {
    // クーポン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
```

```
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getCoupon:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// クーポンコードを指定します
NSString *couponCode = @"0001";
// クーポン取得
[[RKZService sharedInstance]getCoupon:couponCode withBlock:^(RKZCouponData* couponData, RKZResponseStatus *res
ponseStatus) {
  if (responseStatus.isSuccess) {
    // クーポン取得成功
    NSLog(@"code     :%@", couponData.code);
    NSLog(@"name     :%@", couponData.name);
    NSLog(@"image    :%@", couponData.image);
    NSLog(@"image_url :%@", couponData.image_url);
    NSLog(@"possible_from_dte:%@", couponData.possible_from_dte);
    NSLog(@"possible_to_dte  :%@", couponData.possible_to_dte);
    NSLog(@"enable_from_dte  :%@", couponData.enable_from_dte);
    NSLog(@"enable_to_dte  :%@", couponData.enable_to_dte);
    NSLog(@"point       :%@", couponData.point);
  } else {
    // クーポン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

クーポンを交換する

クーポンを交換する場合は、RKZService の exchangeCoupon:couponCd:quantity:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// ユーザーアクセストークンを指定します
NSString *userAccessToken = @"userAccessTokenXXXX";
// クーポンコードを指定します
NSString *couponCode = @"0005";
// クーポン交換枚数を指定します
NSNumber *quantity = @1;
// クーポン交換
[[RKZService sharedInstance]exchangeCoupon:userAccessToken couponCd:couponCode quantity:quantity
                            withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // クーポン交換成功
  } else {
    // クーポン交換失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
getMyCouponList:searchConditionArray:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
NSString *userAccessToken = @"userAccessTokenXXXX";
// マイクーポン取得
[[RKZService sharedInstance] getMyCouponList:userAccessToken searchConditionArray:nil sortConditionArray:nil
                          withBlock:^(NSMutableArray *myCouponDataArray, RKZResponseStatus *responseStatus)
{
  if (responseStatus.isSuccess) {
    // マイクーポン情報取得成功
    for (RKZMyCouponData *myCouponData in myCouponDataArray) {
      NSLog(@"code     :%@", myCouponData.code);
      NSLog(@"coupon_cd    :%@", myCouponData.coupun_cd);
      NSLog(@"coupon_name :%@", myCouponData.coupon_name);
      NSLog(@"get_date   :%@", myCouponData.get_date);
      NSLog(@"use_date   :%@", myCouponData.use_date);
      NSLog(@"used_flg   :%@", myCouponData.used_flg);
      NSLog(@"quanity    :%@", myCouponData.quantity);

      NSLog(@"★CouponData-------------");
      NSLog(@"code     :%@", myCouponData.couponData.code);
      NSLog(@"name     :%@", myCouponData.couponData.name);
      NSLog(@"image      :%@", myCouponData.couponData.image);
      NSLog(@"image_url    :%@", myCouponData.couponData.image_url);
      NSLog(@"possible_from_dte    :%@", myCouponData.couponData.possible_from_dte);
      NSLog(@"possible_to_dte    :%@", myCouponData.couponData.possible_to_dte);
      NSLog(@"enable_from_dte    :%@", myCouponData.couponData.enable_from_dte);
      NSLog(@"enable_to_dte      :%@", myCouponData.couponData.enable_to_dte);
      NSLog(@"point          :%@", myCouponData.couponData.point);
    }
  } else {
    // マイクーポン情報取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の

getMyCoupon:myCouponCd:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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
NSString *userAccessToken = @"userAccessTokenXXXX";
NSString *myCouponCode = @"10";
// マイクーポン取得
[[RKZService sharedInstance]getMyCoupon:userAccessToken myCouponCd:myCouponCode
                          withBlock:^(RKZMyCouponData *myCouponData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // マイクーポン取得成功
    NSLog(@"★MyCouponData-------------");
    NSLog(@"code     :%@", myCouponData.code);
    NSLog(@"coupon_cd    :%@", myCouponData.coupun_cd);
    NSLog(@"coupon_name :%@", myCouponData.coupon_name);
    NSLog(@"get_date   :%@", myCouponData.get_date);
    NSLog(@"use_date   :%@", myCouponData.use_date);
    NSLog(@"used_flg   :%@", myCouponData.used_flg);
    NSLog(@"quanity    :%@", myCouponData.quantity);

    NSLog(@"★CouponData-------------");
    NSLog(@"code     :%@", myCouponData.couponData.code);
    NSLog(@"name     :%@", myCouponData.couponData.name);
    NSLog(@"image      :%@", myCouponData.couponData.image);
    NSLog(@"image_url    :%@", myCouponData.couponData.image_url);
    NSLog(@"possible_from_dte    :%@", myCouponData.couponData.possible_from_dte);
    NSLog(@"possible_to_dte    :%@", myCouponData.couponData.possible_to_dte);
    NSLog(@"enable_from_dte    :%@", myCouponData.couponData.enable_from_dte);
    NSLog(@"enable_to_dte      :%@", myCouponData.couponData.enable_to_dte);
    NSLog(@"point          :%@", myCouponData.couponData.point);
  } else {
    // クーポン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

クーポンを利用する

クーポンを利用する場合は、RKZService の useMyCoupon:myCouponData:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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
NSString *userAccessToken = @"userAccessTokenXXXX";
RKZMyCouponData *myCouponData = [[RKZMyCouponData alloc]init];
myCouponData.code = @"31";
myCouponData.coupun_cd = @"0005";

// クーポン利用
[[RKZService sharedInstance]useMyCoupon:userAccessToken myCouponData:myCouponData
                          withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // クーポン利用成功
  } else {
    // クーポン取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の getPoint:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
//  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
NSString *userAccessToken = @"userAccessTokenXXXX";
// ポイント情報取得
[[RKZService sharedInstance]getPoint:userAccessToken withBlock:^(RKZPointData *pointData, RKZResponseStatus *r
esponseStatus) {
  if (responseStatus.isSuccess) {
    // ポイント情報取得成功
    NSLog(@"point:%@", @(pointData.point));
  } else {
    // ポイント情報情報失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の addPoint:point:contactDate:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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
NSString *userAccessToken = @"userAccessTokenXXXX";
NSNumber *point = @1;
NSDate *contactDate = [NSDate date];
// ポイント加算減算
[[RKZService sharedInstance]addPoint:userAccessToken
                               point:point
                         contactDate:contactDate
                           withBlock:^(RKZPointData *pointData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // ポイント加算減算成功
    NSLog(@"point:%@", @(pointData.point));
  } else {
    // ポイント加算減算失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の getApplicationSettingDataWithBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
 [[RKZService sharedInstance] getApplicationSettingDataWithBlock: ^(RKZApplicationConfigData *applicationConfi
gData, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // アプリケーション設定情報の取得成功
    NSLog(@"point:%@", @(pointData.point));
  } else {
    // アプリケーション設定情報失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getStampRallyList:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
 [[RKZService sharedInstance] getStampRallyList:nil sortConditionArray:nil
                    withBlock:^(NSMutableArray *stampRallyDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリー取得成功
    for (RKZStampRallyData *stampRallyData in stampRallyDataArray) {
      NSLog(@"code :%@", stampRallyData.code);
      NSLog(@"name :%@", stampRallyData.name);
      NSLog(@"short_name :%@", stampRallyData.short_name);
      NSLog(@"stamp_rally_detail :%@", stampRallyData.stamp_rally_detail);
      NSLog(@"stamp_rally_image :%@", stampRallyData.stamp_rally_image);
      NSLog(@"stamp_rally_image_url :%@", stampRallyData.stamp_rally_image_url);
      NSLog(@"stamp_rally_start_date :%@", stampRallyData.stamp_rally_start_date);
      NSLog(@"stamp_rally_end_date :%@", stampRallyData.stamp_rally_end_date);
      NSLog(@"attributes['hoge']  :%@", stampRallyData.attributes[@"hoge"]);
    }
  } else {
    // スタンプラリー取得失敗
```

```
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
getAllStampRallyList:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
 [[RKZService sharedInstance] getAllStampRallyList:nil sortConditionArray:nil
                    withBlock:^(NSMutableArray *stampRallyDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリー取得成功
    for (RKZStampRallyData *stampRallyData in stampRallyDataArray) {
      NSLog(@"code :%@", stampRallyData.code);
      NSLog(@"name :%@", stampRallyData.name);
      NSLog(@"short_name :%@", stampRallyData.short_name);
      NSLog(@"stamp_rally_detail :%@", stampRallyData.stamp_rally_detail);
      NSLog(@"stamp_rally_image :%@", stampRallyData.stamp_rally_image);
      NSLog(@"stamp_rally_image_url :%@", stampRallyData.stamp_rally_image_url);
      NSLog(@"stamp_rally_start_date :%@", stampRallyData.stamp_rally_start_date);
      NSLog(@"stamp_rally_end_date :%@", stampRallyData.stamp_rally_end_date);
      NSLog(@"attributes['hoge']  :%@", stampRallyData.attributes[@"hoge"]);
    }
  } else {
    // スタンプラリー取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode    :%@", @(responseStatus.statusCode));
    NSLog(@"message     :%@", responseStatus.message);
```

```
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の getStampRallySpotList:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
[[RKZService sharedInstance] getStampRallySpotList:nil sortConditionArray:nil
                    withBlock:^(NSMutableArray *stampRallySpotDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリースポット取得成功
    for (RKZStampRallySpotData *stampRallySpotData in stampRallySpotDataArray) {
      NSLog(@"code :%@", stampRallySpotData.code);
      NSLog(@"name :%@", stampRallySpotData.name);
      NSLog(@"stamp_rally_cd :%@", stampRallySpotData.stamp_rally_cd);
      NSLog(@"stamp_rally_name :%@", stampRallySpotData.stamp_rally_name);
      NSLog(@"attributes['hoge']  :%@", stampRallySpotData.attributes[@"hoge"]);
      NSLog(@"spot.code :%@", stampRallySpotData.spot.code);
    }
  } else {
    // スタンプラリースポット取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```


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、RKZService の getStampRallySpotListByStampRallyId:searchConditionArray:sortConditionArray:withBlock :で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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
[[RKZService sharedInstance] getStampRallySpotListByStampRallyId:@"0001" searchConditionArray:nil sortConditio
nArray:nil withBlock:^(NSMutableArray *stampRallySpotDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリースポット取得成功
    for (RKZStampRallySpotData *stampRallySpotData in stampRallySpotDataArray) {
      NSLog(@"code :%@", stampRallySpotData.code);
      NSLog(@"name :%@", stampRallySpotData.name);
      NSLog(@"stamp_rally_cd :%@", stampRallySpotData.stamp_rally_cd);
      NSLog(@"stamp_rally_name :%@", stampRallySpotData.stamp_rally_name);
      NSLog(@"attributes['hoge'] :%@", stampRallySpotData.attributes[@"hoge"]);
      NSLog(@"spot.code :%@", stampRallySpotData.spot.code);
    }
  } else {
    // スタンプラリースポット取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の getStampRallySpotListBySpotId:searchConditionArray:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
[[RKZService sharedInstance] getStampRallySpotListBySpotId:@"0001" searchConditionArray:nil sortConditionArray:nil withBlock:^(NSMutableArray *stampRallySpotDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリースポット取得成功
    for (RKZStampRallySpotData *stampRallySpotData in stampRallySpotDataArray) {
      NSLog(@"code :%@", stampRallySpotData.code);
      NSLog(@"name :%@", stampRallySpotData.name);
      NSLog(@"stamp_rally_cd :%@", stampRallySpotData.stamp_rally_cd);
      NSLog(@"stamp_rally_name :%@", stampRallySpotData.stamp_rally_name);
      NSLog(@"attributes['hoge'] :%@", stampRallySpotData.attributes[@"hoge"]);
      NSLog(@"spot.code :%@", stampRallySpotData.spot.code);
    }
  } else {
    // スタンプラリースポット取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の
stampComplete:stampRallyId:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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。
NSString *userAccessToken = @"userAccessTokenXXXX";
NSString *stampRallyId = @"0001";

// スタンプコンプリート登録
[[RKZService sharedInstance] stampComplete:userAccessToken stampRallyId:stampRallyId
                withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプコンプリート登録成功
  } else {
    // スタンプコンプリート登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
addMyStamp:stampRallyId:spotId:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
// 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。
NSString *userAccessToken = @"userAccessTokenXXXX";
NSString *stampRallyId = @"0001";
NSString *spotId = @"0001";

// 取得スタンプ登録
```

```
[[RKZService sharedInstance] addMyStamp:userAccessToken stampRallyId:stampRallyId spotId:spotId
                withBlock:^(RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // 取得スタンプ登録成功
  } else {
    // 取得スタンプ登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService の
getMyStampHistoryList:searchConditionArray:sortConditionArray:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

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。
NSString *userAccessToken = @"userAccessTokenXXXX";

// スタンプ取得履歴を取得
[[RKZService sharedInstance] getMyStampHistoryList:userAccessToken searchConditionArray:nil sortConditionArra
y:nil withBlock:^(NSMutableArray *myStampHistoryDataArray, RKZResponseStatus *responseStatus) {
  if (responseStatus.isSuccess) {
    // スタンプラリースポット取得成功
    for (RKZMyStampHistoryData *myStampHistoryData in myStampHistoryDataArray) {
      NSLog(@"contact_class_cd :%@", myStampHistoryData.contact_class_cd);
      NSLog(@"stamp_rally_cd :%@", myStampHistoryData.stamp_rally_cd);
      NSLog(@"stamp_rally_name :%@", myStampHistoryData.stamp_rally_name);
      NSLog(@"stamp_rally_spot_cd :%@", myStampHistoryData.stamp_rally_spot_cd);
      NSLog(@"stamp_rally_spot_name :%@", myStampHistoryData.stamp_rally_spot_name);
      NSLog(@"contact_date :%@", myStampHistoryData.contact_date);
```

```
    }
  } else {
    // スタンプ取得履歴取得失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます。
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
addFavoriteToObjectData: userAccessToken:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
//  お気に入り登録にはお知らせ情報とユーザアクセストークンが必要です
RKZObjectData *objectData = [RKZObjectData new];
objectData.object_id = @"object1";
objectData.code = @"0001";
NSString *userAccessToken = @"userAccessTokenXXXX";

// お気に入り登録
[[RKZService sharedInstance] addFavoriteToObjectData: objectData
                             userAccessToken:userAccessToken:
                             ^( RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {

  if (responseStatus.isSuccess) {
    // お気に入り登録成功
  } else {
    // お気に入り登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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、RKZService の
deleteFavoriteToObjectData: userAccessToken:withBlock:で行います。

Blocks により処理結果を返却します。
成功した場合は "responseStatus.isSuccess" が YES , 失敗した場合は NO となります。

```
//  お気に入り削除にはお知らせ情報とユーザアクセストークンが必要です
RKZObjectData *objectData = [RKZObjectData new];
objectData.object_id = @"object1";
objectData.code = @"0001";
NSString *userAccessToken = @"userAccessTokenXXXX";

// お気に入り削除
[[RKZService sharedInstance] deleteFavoriteToObjectData: objectData
                             userAccessToken:userAccessToken:
                             ^( RKZApiStatusCode statusCode, RKZResponseStatus *responseStatus) {

  if (responseStatus.isSuccess) {
    // お気に入り削除成功
  } else {
    // お気に入り登録失敗
    // 失敗時には responseStatus にエラー情報が格納されて返却されます
    NSLog(@"statusCode   :%@", @(responseStatus.statusCode));
    NSLog(@"message    :%@", responseStatus.message);
    NSLog(@"detailMessage:%@", responseStatus.detailMessage);
  }
}];
```

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 RKZService 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
[[RKZService sharedInstance] setDefaultTimeout:10];
RKZResponseStatus *status = [[RKZService sharedInstance] setTenantKey:@"your_tenant_id"];
[[RKZService sharedInstance] getSpotList:nil sortConditionArray:nil withBlock:^(NSMutableArray *spotDataArray,
 RKZResponseStatus *responseStatus) {
    if (responseStatus.isSuccess) {
        NSLog(@"SUCCESS!");
    } else {
        // 10 秒以上かかる場合はエラーとなる
        NSLog(@"statusCode: %@", @(responseStatus.statusCode));
        NSLog(@"message: %@", responseStatus.message);
        NSLog(@"detailMessage: %@", responseStatus.detailMessage);
    }
}];
```

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 RKZService の setTimeouto で行います。

setTimeout メソッドを呼び出すと、指定した指定したタイムアウト時間でタイムアウトする RKZService 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
RKZService *service = [[RKZService sharedInstance] setTimeout:10];
 [[service sharedInstance] getSpotList:nil sortConditionArray:nil withBlock:^(NSMutableArray *spotDataArray, R
KZResponseStatus *responseStatus) {
    if (responseStatus.isSuccess) {
        NSLog(@"SUCCESS!");
    } else {
        // 10 秒以上かかる場合はエラーとなる
        NSLog(@"statusCode: %@", @(responseStatus.statusCode));
        NSLog(@"message: %@", responseStatus.message);
        NSLog(@"detailMessage: %@", responseStatus.detailMessage);
    }
}];
```

※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	2018/02/02	◆ Ver2.1.0 対応版
第 3 版	2019/06/06	◆ Ver2.2.0 対応版
第 4 版	2020/03/09	◆ Ver2.3.0 対応版