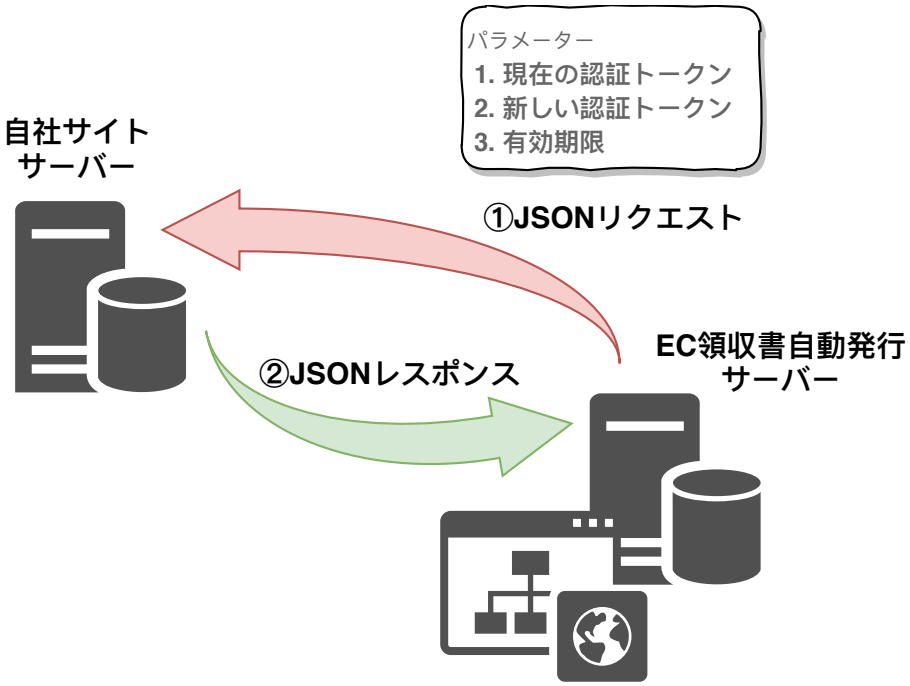
自社サイト注文 API 連携開発ガイド

1 セキュリティ認証

1.1 認証のイメージ



1.2 事前準備

自社サイトのデータベースに以下のトークン保存用テーブルを追加します。

テーブル名：**tamesco_ryoshusho_auth**

カラム名	型	コメント	初期値
tamesco_ryoshusho_token	varchar(100)	セキュリティ認証トークン	""
tamesco_expiration_date	datetime	セキュリティ認証トークンの有効期限（タイムゾーン: Asia/Tokyo）	""

1.3 自社サイトシステムのルート追加

以下のルートを追加します。

> POST /auth

1.4 自社サイトシステムのコントローラー追加

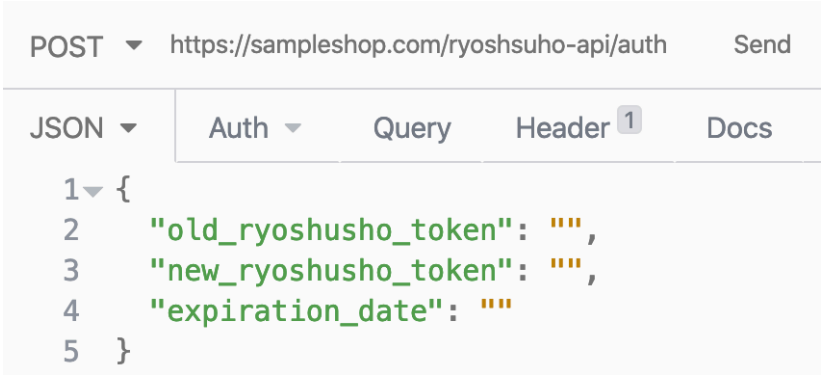
コントローラー（コントローラー名: **TamescoRyoshushoController**）を追加します。

前記ルートに繋げるようコントローラーに **auth()** 関数を追加します。

下記テーブルの内容は「EC領収書自動発行」からの POST リクエストのフォーマット（JSON 文字列）です。リクエストパラメーターを取得し、変数にアサインする必要があります。

パラメーター	値	説明
old_ryoshusho_token	現在のセキュリティ認証トークン	現在利用中のセキュリティ認証トークン
new_ryoshusho_token	新しいセキュリティ認証トークン	更新後の新しいセキュリティ認証トークン
expiration_date	有効期限	新しいセキュリティ認証トークンの有効期限 （タイムゾーン: Asia/Tokyo）

リクエスト例は以下になります。



1.5 auth() 関数の認証トークンに関する処理

auth() 関数でテーブル tamesco_ryoshusho_auth より tamesco_ryoshusho_token 及び tamesco_expiration_date の値を取得します。

tamesco_ryoshusho_token が空文字列の場合は、「認証トークンの初期化処理」を行い、それ以外は「認証トークンの不一致判定・接続確認・更新処理」を行います。

1. **認証トークンの初期化処理**

   tamesco_ryoshusho_token に new_ryoshusho_token の値を保存します。

   tamesco_expiration_date に expiration_date の値を保存します。

   処理完了後、下記の JSON データを返します。

   ```
   {
     "message": "OK",
     "errors": "",
     "data": []
   }
   ```

2. **認証トークンの不一致判定・接続確認・更新処理**

   1. **認証トークンの不一致判定処理**

      tamesco_ryoshusho_token の値と old_ryoshusho_token の値を比較し、一致しない場合は、処理を中止し、下記の JSON データを返します。

      ```
      {
        "message": "",
        "errors": "セキュリティ認証トークン不一致",
        "data": []
      }
      ```

      一致する場合は、処理を継続し以下の処理を行います。

   2. **認証トークンの接続確認・更新処理**

      new_ryoshusho_token が空文字列の場合は、「認証トークンの接続確認」処理を行い、それ以外は「認証トークンの更新」処理を行います。

      1. **認証トークンの接続確認**

         tamesco_expiration_date の値と現在の日付を比較し、認証有効期限切れと判断した場合は、下記の JSON データを返します。

         ```
         {
           "message": "",
           "errors": "セキュリティ認証トークン期限切れ",
           "data": []
         }
         ```

         それ以外の場合は、下記の JSON データを返します。

         ```
         {
           "message": "OK",
           "errors": "",
           "data": []
         }
         ```

2. **認証トークンの更新**

   tamesco_ryoshusho_token に new_ryoshusho_token の値を保存します。

   tamesco_expiration_date に expiration_date の値を保存します。

   保存完了後、下記の JSON データを返します。

```
{
  "message": "OK",
  "errors": "",
  "data": []
}
```

1.6 認証トークンの動作確認

「EC領収書自動発行」 にログインし、「店舗設定」→「API 店舗追加」から店舗の追加を行います。

必須項目を全て入力し、「保存」ボタン押下で、**セキュリティ認証トークン**が自動的に初期化されます。

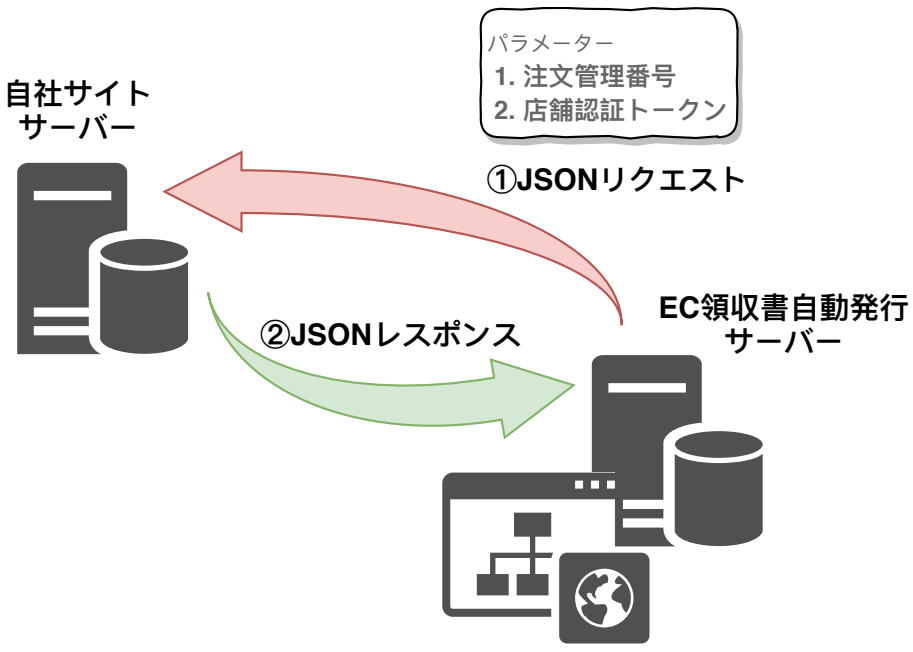
保存後、再度当該店舗の「API 更新」→「セキュリティ認証トークンの更新」押下でエラーが出ないこと且つ「接続確認」ボタン押下でエラーが出ないことを確認します。

以上で認証トークンに関する開発部分が完了します。

認証トークンの更新処理に問題ないかの確認には、tamesco_expiration_date を現在よりも前の日付を設定した状態で、「セキュリティ認証トークンの更新」を押下し、「認証有効期限」が現在の日付に変更されたことを確認します。

2 注文データ取得処理

2.1 注文データ取得のイメージ



2.2 自社サイトシステムのルート追加

以下のルートを追加します。

> POST /order

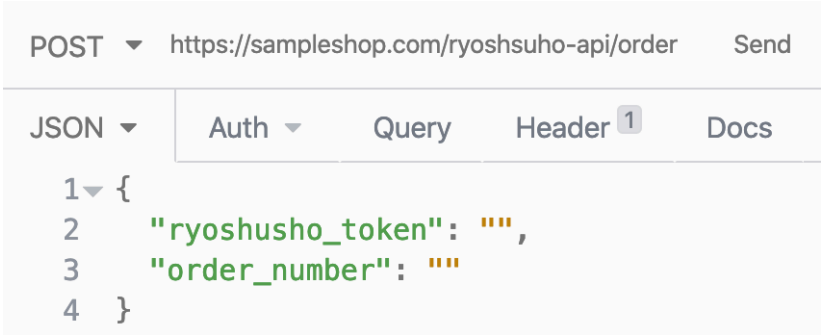
2.3 自社サイトシステムのコントローラー追加

前記ルートに繋げるよう **TamescoRyoshushoController** に **order()** 関数を追加します。

下記テーブルの内容は「EC領収書自動発行」からの POST リクエストのフォーマット（ JSON 文字列）です。リクエストパラメーターを取得し、変数にアサインする必要があります。

パラメーター	パラメーター名	説明
ryoshusho_token	セキュリティ認証トークン	当サービスが発行したセキュリティ認証トークン
order_number	注文管理番号	異なる注文を判定する一意の番号

リクエスト例は以下になります。



2.4 order() 関数の注文データに関する処理

order() 関数でテーブル tamesco_ryoshusho_auth より tamesco_ryoshusho_token 及び tamesco_expiration_date の値を取得します。

tamesco_expiration_date の値と現在の日付を比較し、認証有効期限切れと判断した場合は、下記の JSON データを返します。

```
{
  "message": "",
  "errors": "セキュリティ認証トークン期限切れ",
  "data": []
}
```

認証有効期限切れでない場合は、tamesco_ryoshusho_token の値と old_ryoshusho_token の値を比較し、一致しない場合は、処理を中止し、下記の JSON データを返します。

```
{
  "message": "",
  "errors": "セキュリティ認証トークン不一致",
  "data": []
}
```

前記 2 つのエラーが発生しない場合は、注文データ取得処理を行います。当該注文管理番号の注文データが存在しない場合は、下記の JSON データを返します。

```
{
  "message": "",
  "errors": "注文番号不正",
  "data": []
}
```

当該注文管理番号の注文データが存在する場合は、注文データを返します。注文データのフォーマットは以下の説明を参考してください。

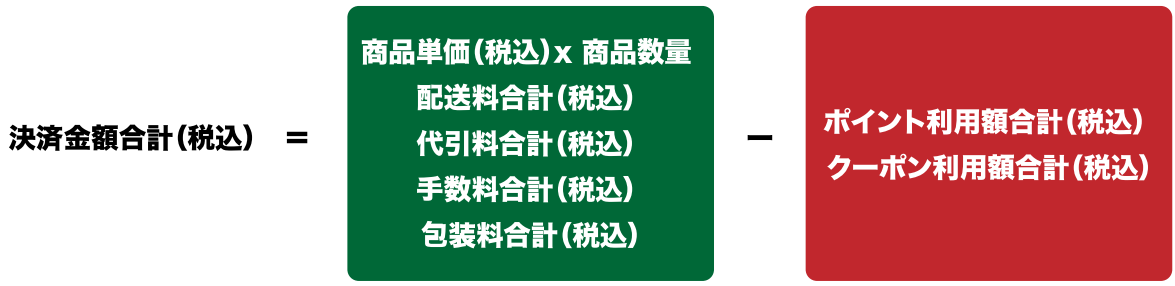
注文データフォーマット

#	SQL項目名	フィールド名	形式	値の設定例
1	order_number	注文管理番号（必須）	varchar(100)	`'sample-100001'` （異なる注文を判定する一意の番号）
2	order_status	注文ステータス（必須）	varchar(100)	`'処理済'` ・ `''` （前記 2 種類以外の値は当サービスでは処理できません。データベース内の値を前記 2 種類の値に変更する必要があります）
3	order_date	注文日（必須）	datetime	`'2020-06-01 15:30:00'`
4	ship_date	出荷日	datetime	`'2020-06-02 15:30:00'`
5	pay_method	決済方法	varchar(100)	`'クレジットカード'` （当サービス店舗設定「発行する決済方法」の文字列と完全一致する必要があります）
6	sender_name	送付先名（必須）	varchar(100)	`'領収 太郎'` （データベースで苗字と名前が別の項目で保存された場合は、1 つの値に変更する必要があります。送付先名がない場合は、注文者名を使用してください）
7	sender_phone	送付先電話番号（必須）	varchar(100)	`'0000'` （個人情報保護の観点で電話番号を半角数字の電話番号下 4 桁のみに変更する必要があります）

8	**sender_zipcode**	**送付先郵便番号（必須）**	varchar(100)	`'0001'` （個人情報保護の観点で郵便番号を半角数字の郵便番号下４桁のみに変更する必要があります）
9	orderer_name	注文者名	varchar(100)	`'領収 太郎'` （データベースで苗字と名前が別の項目で保存された場合は、1つの値に変更する必要があります）
10	orderer_phone	注文者電話番号	varchar(100)	`'0000'` （個人情報保護の観点で電話番号を半角数字の電話番号下4桁のみに変更する必要があります）
11	orderer_zipcode	注文者郵便番号	varchar(100)	`'0000'` （データベースで苗字と名前が別の項目で保存された場合は、1つの値に変更する必要があります）
12	post_price	配送料合計（税込）	int(11)	`'660'` （0以上の整数）
13	pay_charge	手数料合計（税込）	int(11)	`'300'` （0以上の整数）
14	gift_price	包装料合計（税込）	int(11)	`'200'` （0以上の整数）
15	use_point	ポイント利用額合計（税込）	int(11)	`'950'` / `'-950'` （負の値は、当サービスの方で正の値に変更されます）
16	coupon	クーポン利用額合計（税込）	int(11)	`'500'` / `'-500'` （負の値は、当サービスの方で正の値に変更されます）
17	**amount**	**決済金額合計（税込）（必須）**	int(11)	`'3000'` （0以上の整数） この値の定義については以下の「決済金額合計（税込）について」を参照してください。
18	items	注文明細（商品関連情報）	text	JSONデータです。MySQL関数の CONCAT() 、GROUP_CONCAT() 及び JSON_OBJECT() を利用してください。

注文明細（商品関連情報）データフォーマット（items）

「決済金額合計（税込）」はクーポン利用後かつポイント利用額以外の支払い総額である必要があります。



注文明細（商品関連情報）データフォーマット（items）

注文明細（商品関連情報）機能を利用する場合は、以下のすべての情報が必須です。

#	SQL項目名	フィールド名	形式	値の設定例
1	name	商品名（必須）	varchar(100)	`'サンプル商品'`
2	price	商品単価（必須）	int(11)	`'999'`（商品単価が税抜の場合は当サービス店舗設定「**商品単価税率設定**」で設定を変更してください）
3	quantity	商品数量（必須）	int(11)	`'1'`（1以上の整数）
4	tax_ratio	商品税率（必須）	varchar(100)	`0.1` / `10` / `10%`（前記３種類のフォーマットは当サービスですべて処理できます）

JSON サンプルデータ


```
{
  "message":"OK",
  "errors":"",
  "data":{
    "order_number":"SP14721529",
    "order_status":"処理済",
    "order_date":"2020-03-25 17:30:25",
    "ship_date":"2020-03-26 16:30:30",
    "pay_method":"クレジットカード",
    "sender_name":"領収書 太郎",
    "sender_phone":"1234",
    "sender_zipcode":"0001",
    "orderer_name":"タメスコ 太郎",
    "orderer_phone":"5678",
    "orderer_zipcode":"0002",
    "post_price":250,
```

```
    "pay_charge":100,
    "gift_price":100,
    "use_point":450,
    "coupon":0,
    "amount":550,
    "items":[
      {
        "name":"サンプル商品-01",
        "price":10,
        "quantity":5,
        "tax_ratio":0.1
      },
      {
        "name":"サンプル商品-02",
        "price":300,
        "quantity":1,
        "tax_ratio":0.08
      },
      {
        "name":"サンプル商品-03",
        "price":200,
        "quantity":1,
        "tax_ratio":0
      }
    ]
  }
}
```

2.5 注文データ取得処理動作確認

「EC領収書自動発行」 にログインした状態で当該店舗の「領収書発行用 URL」から、データベースに存在する注文を利用し、確認してください。