自社サイト注文 API 連携導入マニュアル

> **必要な動作環境：** 自社サーバーに Web Server + PHP + MySQL の動作環境

上記動作環境以外の場合は、当サービスが提供する **「自社サイト注文 API 関連コード」** で導入することができません。 当サービスが提供する **「注文 API 連携開発ガイド」** に従って開発を行う必要があります。

1 自社サイト注文 API 関連コードのダウンロード

当サービスから **「自社サイト注文 API 関連コード」** をローカル環境に保存・解凍します。

2 自社サイトサーバーに注文 API 関連コードをアップロード

自社サイトサーバー外部公開フォルダに、解凍した注文 API 関連コードをアップロードします。

以下の説明では、

自社サイトトップページ URL	`https://sampleshop.com`
アップロードフォルダ	`ryoshusho-api`

を例として説明します。

3 データベース連携

ブラウザで `https://sampleshop.com/ryoshusho-api` ページを開きます。

以下の初期設定ページ（ `setup.php` ）が表示されることを確認します。

すべての設定項目を入力します。

- **データベース種類***

  MySQL（MariaDB）のみ利用できます。
- **データベース ホスト名***
- **データベース名***
- **データベース ユーザー名***
- **データベース パスワード***
- **注文データ取得コード (SQL) ***

  以下の説明を参考して設定します。

  ただし、設計上、注文管理番号を利用し、データベースで注文を検索します。そのため、注文管理番号のSQL検索値は「**:order_number**」であることが必須です。

```
SELECT * FROM \order\ WHERE \order_number\ = :order_number;
```

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。

決済金額合計（税込）について

「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。

決済金額合計(税込) ＝ **商品単価(税込)x 商品数量**
配送料合計(税込)
代引料合計(税込)
手数料合計(税込)
包装料合計(税込) − **ポイント利用額合計(税込)**
クーポン利用額合計(税込)

注文明細（商品関連情報）データフォーマット（items）

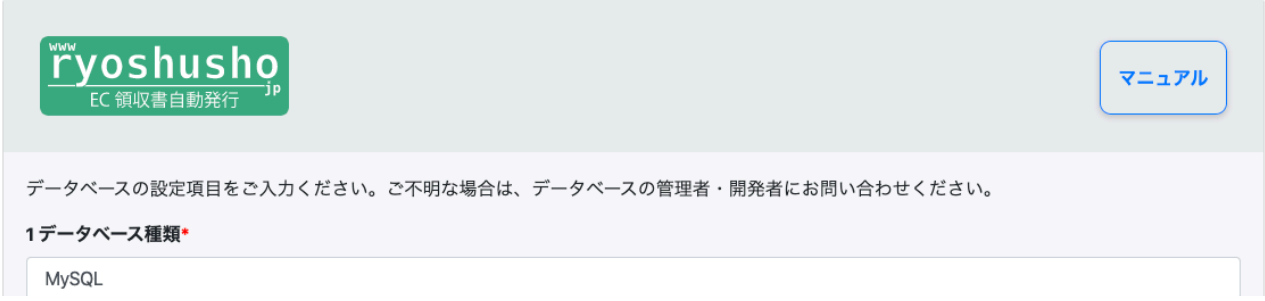
注文明細（商品関連情報）機能を利用する場合は、以下のすべての情報が必須です。

#	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）

（SQL参考例は「**8 ＊SQLコードサンプル**」を参考し、設定してください）

すべての設定項目を入力します。「確認用注文管理番号」でデータベースに存在する番号を入力し、「確認」ボタンで設定に問題がないかの確認ができます。

SQL 文に問題がある場合は、「注文情報」の当該項目は赤色で表示されます。



2 データベース ホスト名*

mysql.sampleshop.com

localhost以外の場合は、実際のホスト名に変更してください。

3 データベース名*

sampleshop

注文情報を保存するデータベースの名前をご入力ください。

4 データベース ユーザー名*

admin

5 データベース パスワード*

••••••

6 注文データ取得コード (SQL)*

```
SELECT
  `order_number` AS order_number,
  IF(`order_status`='処理済' or `order_status` = '発送済', '処理済', '') AS order_status,
  `order_date` AS order_date,
  `ship_date` AS ship_date,
  `pay_method` AS pay_method,
  CONCAT(`sender_first_name`, ' ', `sender_last_name`) AS sender_name,
  RIGHT(`sender_phone`, 4) AS sender_phone,
  RIGHT(`sender_zipcode`, 4) AS sender_zipcode,
  CONCAT(`orderer_first_name`, ' ', `orderer_last_name`) AS orderer_name,
  RIGHT(`orderer_phone`, 4) AS orderer_phone,
  RIGHT(`orderer_zipcode`, 4) AS orderer_zipcode,
  `post_price` AS post_price,
  `pay_charge` AS pay_charge,
  `gift_price` AS gift_price,
  `use_point` AS use_point,
  `coupon` AS coupon,
  `amount` AS amount,
  `items` AS items
```

確認用注文管理番号

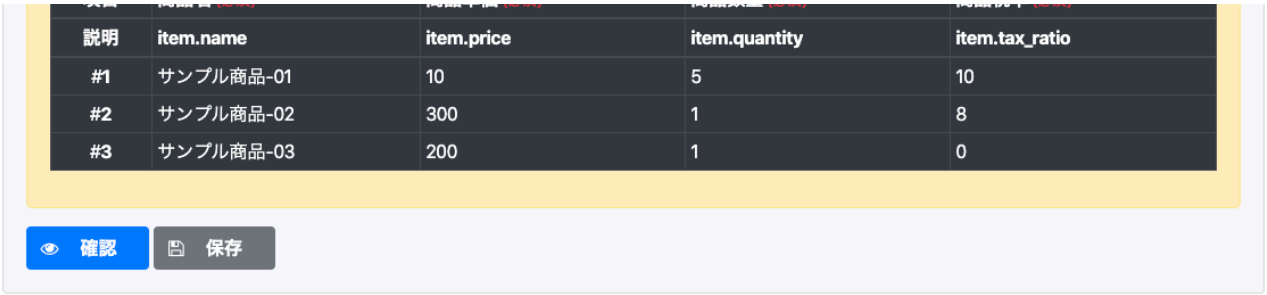
SP14721529

注文情報

項目	説明	取得値
order_number	注文管理番号 (必須)	SP14721529
order_status	注文ステータス (必須)	処理済
order_date	注文日 (必須)	2020-03-25 17:30:25
ship_date	出荷日	2020-03-26 16:30:30
pay_method	決済方法	クレジットカード
sender_name	送付先名 (必須)	領収書 太郎
sender_phone	送付先電話番号 (必須)	1234
sender_zipcode	送付先郵便番号 (必須)	0001
orderer_name	注文者名	タメスコ 太郎
orderer_phone	注文者電話番号	5678
orderer_zipcode	注文者郵便番号	0002
post_price	配送料合計（税込）	250
pay_charge	手数料合計（税込）	100
gift_price	包装料合計（税込）	100
use_point	ポイント利用額合計（税込）	450
coupon	クーポン利用額合計（税込）	0
amount	発行金額合計（税込） (必須)	550
items	注文明細（商品関連情報） (JSON)	注文明細（商品関連情報）は次の表でご確認ください

注文明細（商品関連情報）

項目	商品名 (必須)	商品単価 (必須)	商品数量 (必須)	商品税率 (必須)



初期設定に問題がないことを確認し、「保存」ボタン押下で設定を保存します。また、再設定時に便利な**【領収書】API設定情報.txt** が自動的にダウンロードされます。データベースにアクセスできる情報が入っているため、必ず安全な場所に保存してください。

また、「**注文情報取得API**」ページが表示されることを確認します。

「接続状態」が「設定完了・未確認」の表示になっていることを確認し、**API-URL**をコピーします。（当サービスの API 店舗追加時に必要です）

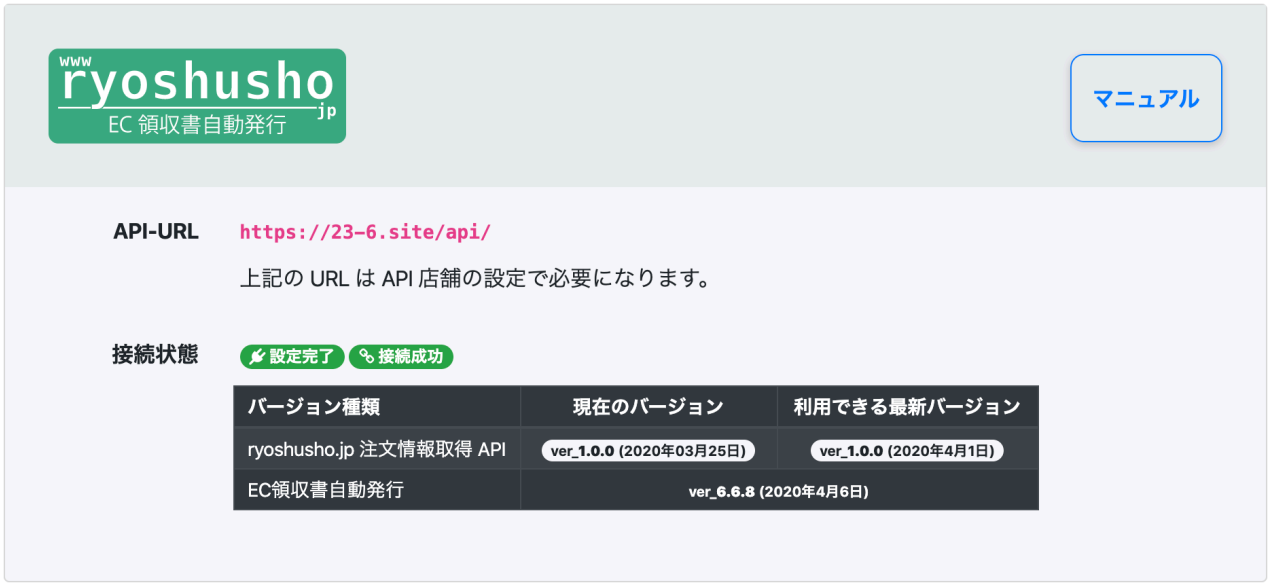
4 当サービスの API 店舗追加

「EC領収書自動発行」 にログインして、「店舗設定」→「API 店舗追加」を押下し、**「事前準備が完了しましたか。」**で「はい」を選択すると、店舗追加設定を開始します。

「店舗追加」の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。

なお、**認証有効期限** が **90日** です。

ブラウザで `https://sampleshop.com/ryoshusho-api` ページを開き、「接続成功」が表示されることを確認します。



以上で、自社サイト注文API連携導入手続が完了になります。

5 セキュリティ認証トークンの更新（参考情報）

セキュリティ認証トークンには 90 日の認証有効期限があります。

有効期限 1 営業日前に、当サービスよりメールで更新依頼を送信します。 **「EC領収書自動発行」** をログインした状態で「店舗設定」→ 当該店舗の「API 更新」→「セキュリティ認証トークンの更新」ボタンを押します。

「**認証有効期限**」が更新されることを確認し、「保存」ボタンを押します。

6 注文 API 関連コードの再設定（参考情報）

当社が提供する注文 API 関連コードには変更機能がありません。注文 API 関連コードを再設定が必要な場合は、以下の手順で再設定するか、`ryoshusho-api` フォルダ内の `order.php` を直接編集する必要があります。

1. 自社サイトサーバーにある `ryoshusho-api` フォルダ内の全てのファイルを削除
2. 本マニュアル「**1 自社サイト注文 API 関連コードのダウンロード**」から
   「**3 データベース連携**」までの設定を行う
3. **「EC領収書自動発行」** をログインした状態で「店舗設定」→ 当該店舗の「詳細編集」→
   「**セキュリティ認証トークンの初期化**」ボタンを押下

7 データベースパスワードが定期的に変更される場合（参考情報）

前記「**6 注文 API 関連コードの再設定（参考情報）**」の手順を実施するか、`ryoshusho-api` フォルダ内の `order.php`（**DB_PASSWORD** 部分）を直接編集する必要があります。

ただし、`order.php` には注文データを取得するための重要設定が入っているため、編集する場合は、バックアップをした上、慎重に行ってください。

8 ＊SQLサンプルコード（参考情報）

```
SELECT
    `order_number` AS order_number,
    IF(`order_status`='処理済' or `order_status` = '発送済', '処理済', '') AS
order_status,
    `order_date` AS order_date,
    `ship_date` AS ship_date,
    `pay_method` AS pay_method,
    CONCAT(`sender_first_name`, ' ', `sender_last_name`) AS sender_name,
    RIGHT(`sender_phone`, 4) AS sender_phone,
    RIGHT(`sender_zipcode`, 4) AS sender_zipcode,
    CONCAT(`orderer_first_name`, ' ', `orderer_last_name`) AS orderer_name,
    RIGHT(`orderer_phone`, 4) AS orderer_phone,
    RIGHT(`orderer_zipcode`, 4) AS orderer_zipcode,
    `post_price` AS post_price,
    `pay_charge` AS pay_charge,
    `gift_price` AS gift_price,
    `use_point` AS use_point,
    `coupon` AS coupon,
    `amount` AS amount,
    `items` AS items
FROM
    `order`,
    (
        SELECT
            CONCAT('[', GROUP_CONCAT(JSON_OBJECT('name', name, 'price', price,
'quantity', quantity, 'tax_ratio', tax_ratio)), ']') AS items
        FROM
            `order_detail`
        WHERE
            `order_number` = :order_number
    ) AS items
WHERE
    `order_number` = :order_number;
```

データベースのカラム名、値については「**注文データフォーマット**」、「**注文明細（商品関連情報）データフォーマット（items）**」で確認してください。