ＰＧマルチペイメントサービス

トークン決済　サービス仕様書

2023 年 11 月 21 日 1.39 版

☑ 本書の著作権は「GMO ペイメントゲートウェイ株式会社(以下、弊社という)」に帰属します。

☑ 本書のいかなる部分においても、弊社に事前の書面による許可なく、電子的、機械的を含むいかなる手段や形式によってもその複製、改変、頒布、ならびにそれらに類似する行為を禁止します。

☑ 本仕様書は加盟店様と弊社との間で締結した機密保持契約において機密情報として規定される情報です。本仕様書の取扱いは機密保持契約の規定に従ってください。

☑ 本書の内容は、予告なく随時更新されます。

☑ 本書の内容について万全を期しておりますが、万一記載もれ等お気づきの点がございましたら、弊社までご連絡ください。

目次

変更履歴

2014 年 11 月 20 日 –draft-1

- 新規作成

2015 年 1 月 20 日 1.00

- 加盟店検証環境リリース版

2015 年 1 月 28 日 1.01

- エラーコードに関する誤記を修正　100 番が重複していたものを訂正/不要なコードを削除
- サポートブラウザに関する情報を追記　InternetExplorer について、8 以降の標準モードのみサポート

2015 年 6 月 1 日 1.02

- JavaScript サンプルソースを一部修正

2015 年 8 月 26 日 1.03

- フォームを一部修正、文言追記

2016 年 5 月 23 日 1.04

- 「1. はじめに」　の不要文言削除

2016 年 8 月 22 日 1.05

- 「2.1. トークン決済の特徴」に多通貨クレジットカード決済の文言を追加
- 「3.3.1. トークンを利用して、マルチペイメントの決済/会員登録 API を実行する」に対象の仕様書を記載
- 「4.2. サポートされるブラウザ環境について」の文言を修正
- 「4.3. 動作確認済みの OS／ブラウザについて」を追加
- 「5.3. モジュールタイプ(php)　ExecTran 呼び出しパラメータ」を削除
- 「5.4. モジュールタイプ(java)　ExecTran 呼び出しパラメータ」を削除

2016 年 9 月 20 日 1.06

- 「2.1. トークン決済の特徴」にカード登録／更新の文言を追加
- 「3.3.1. トークンを利用してマルチペイメントサービスの API を実行する」にカード登録／更新の文言を追加

2017 年 5 月 8 日 1.07

- 「3.2.4. フォーム全体のイメージ」の expire 部分変更

2017 年 7 月 18 日 1.08

- 「3.2.4. フォーム全体のイメージ」に tokennumber を追加
- 「5.2.2. getToken」に tokennumber の説明を追加
- 「5.2.3. getToken コールバック」の token の説明を修正

2017 年 8 月 21 日 1.09

- サンプルコードの誤り修正
- トークン化するカードのカード名義人に-(ハイフン)、/(スラッシュ)の追加
- 「4. API 版」の追加
- 仕様書構成の変更

2017 年 9 月 19 日 1.10

- 「2.1. トークン決済の概要」に説明を追記
- 「4.2.5. サンプルコード」を追加
- 「5.3. 処理結果コード表」に説明文言修正

2017 年 11 月 13 日 1.11

- 「5.3. 処理結果コード」の不要エラーコードを削除

2018 年 1 月 15 日 1.12

- 動作確認済みの OS／ブラウザを最新のデータに更新
- 「JavaScript」の表記ゆれを修正

2018 年 3 月 12 日 1.13

- 「3.2.3.3. コールバック変数について」を追記
- 「5.3. 処理結果コード表」に 902 を追加

2018 年 4 月 16 日 1.14

- 動作確認済みの OS／ブラウザを最新のデータに更新

2018 年 5 月 14 日 1.15

- 動作確認済みの OS／ブラウザを最新のデータに更新

2018 年 7 月 17 日 1.16

- 動作確認済みの OS／ブラウザを最新のデータに更新

2018 年 8 月 13 日 1.17

- 「3.2.2.2. getToken」「3.2.4.4. フォーム全体のイメージ」「4.2.3. 入力パラメータ(スマートフォンアプリ⇒当サービス)へ」に文言追記
- 「5.2. トークンの有効期限について」の文言を一部変更

2018 年 10 月 15 日 1.18

- 動作確認済みの OS／ブラウザを最新のデータに更新

2019 年 1 月 15 日 1.19

- 動作確認済みの OS／ブラウザを最新のデータに更新

2019 年 2 月 12 日 1.20

- 3.2.1.「トークン取得 JavaScript URL」に新 URL の記載を追加
- 5.3.「処理結果コード表」に 701 を追加

2019 年 4 月 15 日 1.21

- 動作確認済みの OS／ブラウザを最新のデータに更新
- 「5.1.1. トークンを利用してマルチペイメントサービスの API を実行する」のカード登録／更新の場合の置き換え対象項目に SecurityCode を追加

2019 年 7 月 16 日 1.22

- 動作確認済みの OS／ブラウザを最新のデータに更新
- 「5.1.1. トークンを利用してマルチペイメントサービスの API を実行する」のカード登録／更新の場合の SecurityCode 置き換え機能の利用開始日を変更

2019 年 8 月 19 日 1.23

- 「5.1.1. トークンを利用してマルチペイメントサービスの API を実行する」のカード登録／更新の場合の SecurityCode 置き換え機能の利用開始日を削除
- 「4.2.5.2. iOS (Swift 版)」のサンプルコードを一部修正
- 「多通貨クレジットカード決済」から「多通貨クレジットカード決済（MCP)」に表記を変更

2019 年 9 月 17 日 1.24

- 「多通貨クレジットカード決済（MCP)」から「多通貨クレジットカード決済（MCP)、多通貨クレジットカード決済（DCC)」に表記を変更

2019 年 10 月 15 日 1.25

- 動作確認済みの OS／ブラウザを最新のデータに更新

2020 年 1 月 20 日 1.26

- 動作確認済みの OS／ブラウザを最新のデータに更新

2020 年 4 月 14 日 1.27

- 動作確認済みの OS／ブラウザの注記に日時設定を追加

2020 年 7 月 21 日 1.28

- callback 関数名に関する許容文字列およびエラーコードの追加
- トークン取得 JavaScript URL の旧 URL を削除
- 動作確認済みの OS／ブラウザを最新のデータに更新

2020 年 10 月 20 日 1.29

- 動作確認済みの OS／ブラウザを最新のデータに更新

2021 年 1 月 19 日 1.30

- 動作確認済みの OS／ブラウザを最新のデータに更新

2021 年 4 月 20 日 1.31

- 動作確認済みの OS／ブラウザを最新のデータに更新

2021 年 7 月 20 日 1.32

- 動作確認済みの OS／ブラウザを最新のデータに更新

2021 年 10 月 19 日 1.33

- 動作確認済みの OS／ブラウザを最新のデータに更新

2022 年 1 月 18 日 1.34

- 動作確認済みの OS／ブラウザを最新のデータに更新

2022 年 4 月 19 日 1.35

- 動作確認済みの OS／ブラウザを最新のデータに更新

2022 年 9 月 21 日 1.36

- 動作確認済みの OS／ブラウザを最新のデータに更新

2023 年 5 月 23 日 1.37

- 動作確認済みの OS／ブラウザを最新のデータに更新

2023 年 6 月 20 日 1.38

- サービス提供を終了したため、多通貨クレジットカード決済（MCP）の記載を削除

2023 年 11 月 21 日 1.39

- 「5.1.1. トークンを利用してマルチペイメントサービスの API を実行する」のカード決済インターフェースは Web マニュアルに移行

1. はじめに

当資料は、マルチペイメントのトークン方式に関する説明資料です。

トークン発行に関係する箇所のみ記載されています。その他、通常の決済の流れは、各接続方式(プロトコルタイプ/モジュールタイプ)マニュアルをご覧ください。

本ドキュメントに例示されたコード例について、一切の責任を負いません。実装に当たっては、加盟店様にて十分な検証をいただくよう、お願いします。

2. トークン決済の概要

2.1. トークン決済の特徴

トークン決済は、加盟店様の決済に以下の特徴をもたらす接続方式です。

・カード番号完全非保持

→データベース等の永続的な保持はもとより、加盟店様サーバーを通過する全ての電文においてカード番号に触れず
クレジットカード決済、カード登録／更新、多通貨クレジットカード決済（DCC）が実装可能です。

・自由なサイト構成

→加盟店様 web サイト上に、JavaScript によるカード番号トークン化処理を実装していただきます。
また、アプリ向けには、API 版も提供しております。
決済は加盟店様とマルチペイメントのサーバー間で実行します。これにより、加盟店様の画面遷移が自由に設計可能です。

以下の利用を推奨しております。

JavaScript 版	ブラウザ、スマートフォンアプリの WebView
API 版	スマートフォンのネイティブアプリ ※ブラウザで API 版を利用した場合、カード番号完全非保持は実現できませんので、JavaScript 版をご利用ください。

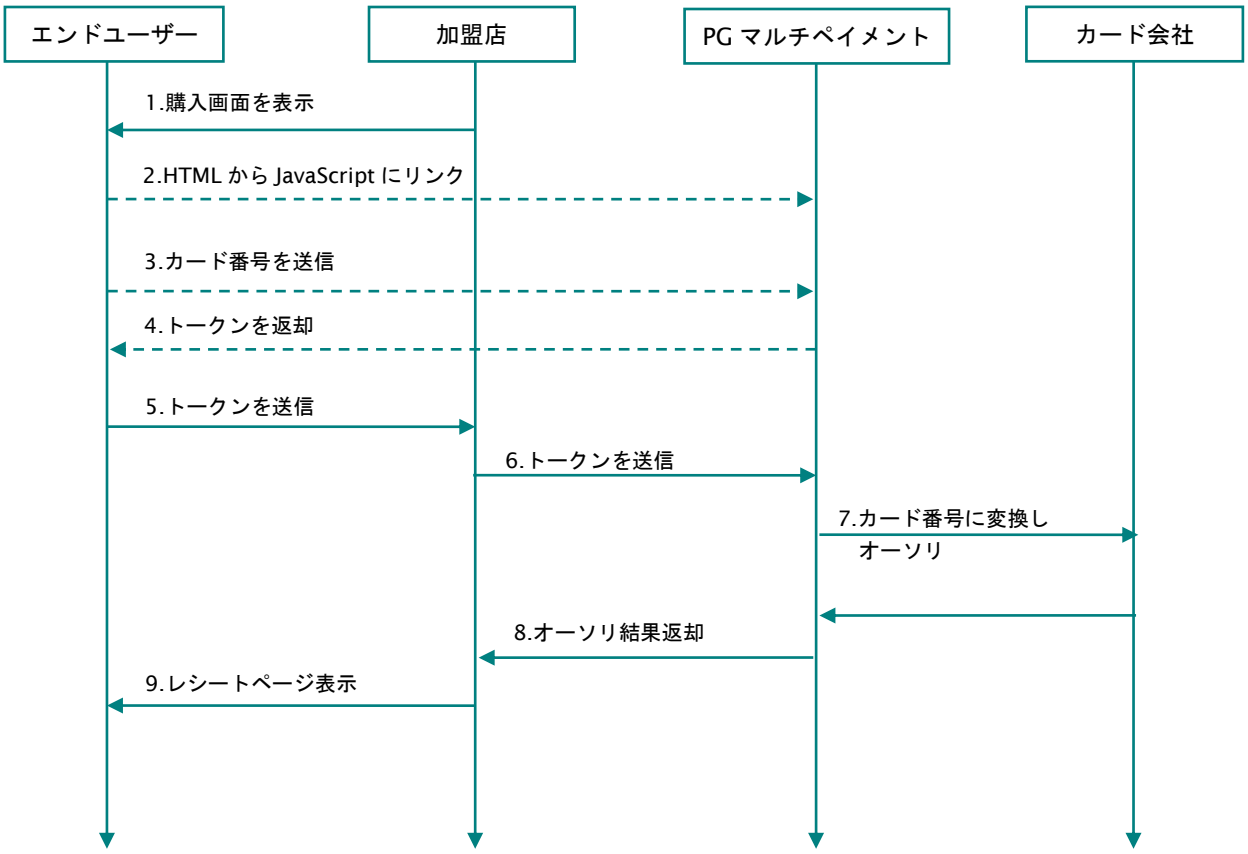
3. JavaScript 版

3.1. 決済の流れ

3.1.1. 決済の流れとシステムシーケンス

トークン決済の流れを、マルチペイメントの決済 API の流れとあわせて、以下に記述します。

図 3.1.1-1 決済イメージ



3.2. リファレンス

3.2.1. トークン取得 JavaScript URL

トークン取得のための JavaScript は、以下の URL をリンクしてお使いください。

テスト環境と本番環境で URL が異なります。ご注意ください。

No	URL
テスト環境	https://stg.static.mul-pay.jp/ext/js/token.js
本番環境	https://static.mul-pay.jp/ext/js/token.js

3.2.2. Multipayment オブジェクト　メソッド API

3.2.2.1. init

Multipayment オブジェクトを利用する際、オブジェクトのロード後に必ず呼び出すメソッドです。

メソッドシグニチャ

```
Multipayment.init( apiKey )
```

呼び出しパラメータ

名称	意味	設定する値
apiKey	Multipayment を利用するための ApiKey です。	マルチペイメントサービスから発行された、ショップ ID を設定してください。

3.2.2.2. getToken

トークン取得を行うメソッドです。取得結果は、callback パラメータで指定した関数に JavaScript オブジェクトとして渡されます。

発行されたトークンは、有効期限が経過するか、一度 API で利用されると、無効となります。

複数の API でトークンを利用される場合は、tokenNumber にてトークンを複数発行してください。

メソッドシグニチャ

```
Multipayment.getToken( cardObject , callback )
```

パラメータ

名称	意味	設定する値
cardObject	以下のプロパティを持つ、JavaScript オブジェクトです。	加盟店様の入力フォーム等で取得したカード情報を設定してください。
callback	トークン取得後にコールバックする、JavaScript function です。	コールバック対象の JavaScript 関数オブジェクトを設定してください。 ※無名関数はご利用いただけません。 ※関数名指定の許可文字種は以下の通りです。 半角英数字（a-zA-Z0-9） アンダースコア（_） ピリオド（.）

cardObject

名称	必須	意味	設定する値
cardno	◎	トークン化するカード番号	カード番号を、ハイフンなし/半角数字で設定してください。
expire	◎	トークン化するカードの有効期限	カード有効期限を、半角数字で YYMM 又は YYYYMM 形式いずれかで設定してください。
securitycode		トークン化するカードのセキュリティコード	セキュリティコードを、3 又は 4 桁の数字で設定してください。設定されない場合、セキュリティコードを利用せずに決済を行います。
holdername		トークン化するカードのカード名義人	カード名義人を、以下の文字種で 50 文字以内で設定してください。設定されない場合、カード名義人を取引に記録しません。 文字種：(全て半角) 数字/アルファベット(大文字/小文字)/ “ “（スペース）,(コンマ）.(ピリオド）-(ハイフン)/(スラッシュ)
tokennumber		トークンを発行する数	発行するトークンの数を 1～10 の数値で設定してください。 入力値無しの場合は、1 個のトークンを発行します。

3.2.2.3. getToken コールバック

Multipayment.getToken()の処理後に呼び出される、コールバック関数です。

メソッドシグニチャ

[getToken で指定したコールバック関数](result)

コールバックの result には、以下の JavaScript object が設定されます。

名称		意味	設定する値
resultCode		処理結果を表すコード	別表の結果コードが文字列で設定されます。複数のエラーが発生した場合、最初の 1 つが設定されます。
tokenObject		取得したトークン情報 以下のプロパティを持つオブジェクト	トークン情報が、JavaScript object で設定されます。なお、resultCode が"000"(取得完了)以外の場合、このオブジェクト自身もしくはプロパティが設定されない場合があります。
	token	トークン文字列	トークン文字列です。決済時には、この値を設定してください。
		トークン配列 ※1	tokennumber に値が設定されていた場合は、トークン文字列を配列にした値が設定されます。
	toBeExpiredAt	トークン有効期限	トークンの有効期限です。 yyyy-mm-dd-HH-MM-SS 形式で設定されます。
	maskedCardNo	マスク済みカード番号	トークン化されたカード番号が、一部*マスクされて設定されます。
	isSecurityCodeSet	セキュリティコード設定フラグ	トークン取得時にセキュリティコードが設定された事を表すフラグです。bool 値で設定され、true の場合セキュリティコード設定ありとされます。

※1.result サンプル(tokennumber=3)

{"tokenObject":{"token":["a33c8bec609ffc75726249d8d82286d529bd1deb973119cf497eeb54610ab9d2","c89d7a6484af2b116e9a8636f8c1020de885bf2765a3ab3565f20f029dcc3150","719bfb8ec852c0b56a4cde92336e7f0a142a8e25f9018f3f1ec129817b87bc98"],"toBeExpiredAt":"2016-09-26-17-56-38","maskedCardNo":"411111******111","isSecurityCodeSet":true},"resultCode":"000"}

3.2.3. JavaScript 版注意事項

3.2.3.1. サポートされるブラウザ環境について

JavaScript 版トークン方式は、JavaScript の動作するブラウザを前提としております。JavaScript の利用できない端末(一部フィーチャーフォン等)では、トークン方式をサポートいたしません。

3.2.3.2. 動作確認済みの OS／ブラウザについて

JavaScript 版は以下の表の〇で記載した OS／ブラウザにて動作確認を行っております。

空欄は弊社が動作確認を行っていない環境です。

表 3.2.3.2-1　OS／ブラウザ別動作確認表（Windows）

	Google Chrome				Mozilla Firefox			Microsoft Edge		
	109	110	111	112	109	110	111	109	110	111
Windows 10	〇	〇	〇	〇	〇	〇	〇	〇	〇	〇
Windows 11	〇	〇	〇	〇	〇	〇	〇	〇	〇	〇

表 3.2.3.2-2　OS／ブラウザ別動作確認表（Mac）

ブラウザ / OS	Google Chrome				Mozilla Firefox			Safari
	109	110	111	112	109	110	111	13.1
macOS Catalina	〇	〇	〇	〇	〇	〇	〇	〇

表 3.2.3.2-3　OS／ブラウザ別動作確認表（iOS）

ブラウザ / OS	Google Chrome	Safari
iOS 15.6	〇	〇
iOS 15.7	〇	〇
iOS 16.1	〇	〇
iOS 16.2	〇	〇
iOS 16.3	〇	〇

表 3.2.3.2-4　OS／ブラウザ別動作確認表（Android）

ブラウザ / OS	Google Chrome	Mozilla Firefox
	112	111
Android 9 pie	○	○
Android 10	○	○
Android 11	○	○
Android 12	○	○
Android 13	○	○

※ 動作を確認していない OS／ブラウザでも利用可能な場合があります。

※ 動作確認済みの OS／ブラウザをご利用の場合でも、
エンドユーザのご利用環境（設定、日時設定、通信速度等）によっては正しく利用できない場合があります。

※ Internet Explorer は、version 11 以降の標準モードのみサポートされます。
（Quirks モードでは動作いたしません。）

※ 動作環境は随時検証したうえ更新されます。なお一度掲載したバージョンにおいて、メーカーのサポートが終了した場合や新しいバージョンがリリースした場合、記載から削除される場合があります。

3.2.3.3.　コールバック変数について

コールバック関数はグローバルで宣言する必要があります。

コールバック関数は関数名を宣言してください。無名の場合、ブラウザによって動作しない場合があります。

以下のグローバル変数を使用しております。

Multipayment

CryptoJS

JSEncryptExports

JSEncrypt

ASN1

Base64

Hex

KJUR

3.2.4. 購入画面の JavaScript 組み込み

3.2.4.1. JavaScript を読み込む

加盟店様の購入ページ上で、JavaScript へのリンクを記述してください。

・・・

<script src="https://stg.static.mul-pay.jp/ext/js/token.js"></script>

・・・

※URL はマルチペイメントテスト環境/本番環境で異なります。本ドキュメントのリファレンスをご確認ください。

3.2.4.2. マルチペイメントオブジェクトを利用してトークン化

マルチペイメントオブジェクトを利用して、トークンを取得します。カード情報と、トークン取得後に実行する処理を指定します。

```
―――――
<script type=" text/javascript" >
  Multipayment.init( 'tshop00000001');      ・・・トークンを利用するための api キー
  Multipayment.getToken(
   {
    cardno: '4111111111111111',             ・・・加盟店様の購入フォームから取得したカード番号
    expire: '201501',                       ・・・加盟店様の購入フォームから取得したカード有効期限
    securitycode:' 111',                    ・・・加盟店様の購入フォームから取得したセキュリティコード
    holdername:' SOME HOLDER',              ・・・加盟店様の購入フォームから取得したカード名義人
    tokennumber:' 5'                        ・・・加盟店様の購入フォームから取得したトークン発行数
   },someCallbackFunction                   ・・・トークン取得後に実行する javascript function
  );
</script>
―――――
```

3.2.4.3. トークン取得後に、購入フォームを加盟店様に送信

トークン取得後の callback に指定した function 定義です。この中で加盟店様の購入フォームを submit します。

callback は、トークン取得結果オブジェクトが引数に渡されます。

```
―――――
function someCallbackFunction ( response ){
  if( response.resultCode != ‘000’ ){
    window.alert( ‘購入処理中にエラーが発生しました’ )
  }else{
    //カード情報は念のため値を除去
    document.getElementById( ‘cardno’ ).value=”” ;
    document.getElementById( ‘expire’ ).value=””
    document.getElementById( ‘securitycode’ ).value=””
    document.getElementById( ‘tokennumber’ ).value=””
    //予め購入フォームに用意した token フィールドに、値を設定
    document.getElementById( ‘token’ ).value = response.tokenObject.token
    //スクリプトからフォームを submit
    document. getElementById( ‘purchaseform’ ).submit()
  }
}
―――――
```

3.2.4.4. フォーム全体のイメージ

購入フォーム全体のイメージは以下の通りです。

```
<html>
<head>
<script src="https://stg.static.mul-pay.jp/ext/js/token.js" ></script>
</head>
<body>
        <!-- カード情報トークン化フォーム -->
        <form id="getTokenForm">
        <p>カード番号：<input type="text" value="" name="cardno" id="cardno" /></p>
        <p>カード有効期限：<input type="text" value="" name="expire_year" id="expire_year" />/
        <input type="text" value="" name="expire_month" id="expire_month" />
        </p>
        <p>カード名義人：<input type="text" value="" name="holdername" id="holdername" /></p>
        <p>セキュリティコード：<input type="text" value="" name="securitycode" id="securitycode" /></p>
        <p>発行数：<input type="text" value="" name="tokennumber" id= "tokennumber" /></p>
        <p><input type="button" value="購入する"  onclick="doPurchase()" /></p>
        </form>
        <!-- 加盟店購入フォーム -->
        <form id="purchaseForm" action="(加盟店様 URL)" method="post">
        <p>
        <input type="hidden" value="" name="kameitenn_chumon_bango" /> <!-- 加盟店での注文番号等、決済を特定できる情報 -->
        <input type="hidden" value="" id="token" name="token" /> <!-- 取得したトークンを設定するプレースホルダ -->
        </p>
        </form>
        <script type="text/javascript">
                function execPurchase(response) {
                        if (response.resultCode != "000") {
                                window.alert("購入処理中にエラーが発生しました");
                        } else {
                                // カード情報は念のため値を除去
                                document.getElementById("cardno").value = "";
                                document.getElementById("expire_year").value = "";
                                document.getElementById("expire_month").value = "";
                                document.getElementById("securitycode").value = "";
                                document.getElementById("tokennumber").value = "";
                        // 予め購入フォームに用意した token フィールドに、値を設定
                        //発行されたトークンは、有効期限が経過するか、一度 API で利用されると、無効となります。
                        //複数の API でトークンを利用される場合は、tokenNumber にてトークンを複数発行してください。
                                document.getElementById("token").value = response.tokenObject.token;
```

```
                                        // スクリプトからフォームを submit
                                        document.getElementById("purchaseForm").submit();
                                }
                        }

                        function doPurchase() {
                                var cardno, expire, securitycode, holdername;
                                var cardno = document.getElementById("cardno").value;
                                var expire =
                               document.getElementById("expire_year").value + document.getElementById("expire_month").value;
                                var securitycode = document.getElementById("securitycode").value;
                                var holdername = document.getElementById("holdername").value;
                                var tokennumber = document.getElementById("tokennumber").value;
                                Multipayment.init("tshop00000001");
                                Multipayment.getToken({
                                        cardno : cardno,
                                        expire : expire,
                                        securitycode : securitycode,
                                        holdername : holdername,
                                        tokennumber : tokennumber
                                }, execPurchase);
                        }
</script>
</body>
</html>
```

4. API 版

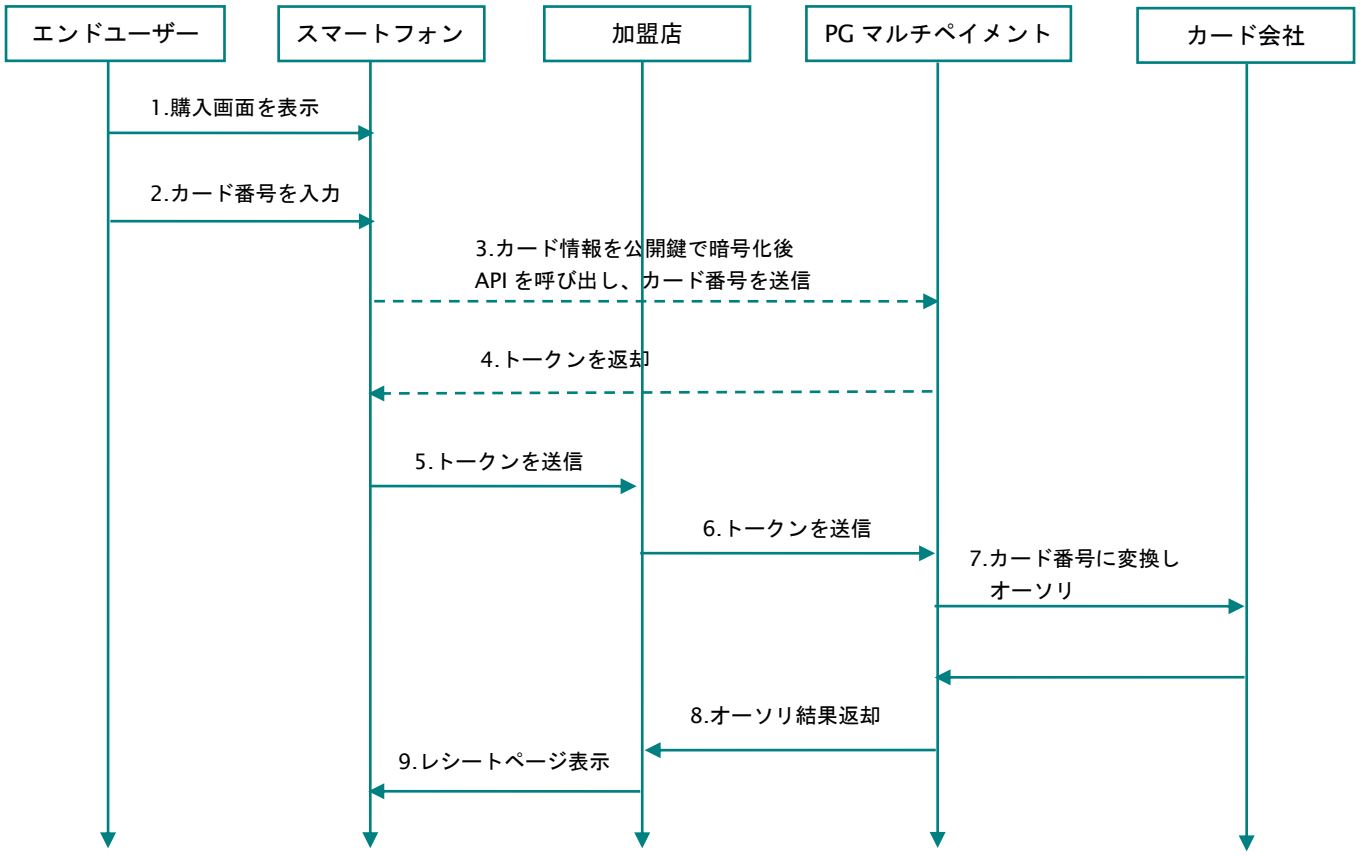
ブラウザ経由の場合は JavaScript 版をご利用ください。

4.1. 決済の流れ

4.1.1. 決済の流れとシステムシーケンス

トークン決済の流れを、マルチペイメントの決済 API の流れとあわせて、以下に記述します。

図 4.1.1-1 決済イメージ



4.2. リファレンス

4.2.1. インタフェース仕様

ここでは､インタフェース詳細について説明します｡インタフェースの呼び出しは HTTPS 通信にて行う必要があります。

4.2.2. 接続先 URL

/ext/api/credit/getToken

4.2.3. 入力パラメータ(スマートフォンアプリ⇒当サービス)

各パラメータ値を POST で送信します。

HTTP リクエストの Content-Type ヘッダの MIME タイプは「application/x-www-form-urlencoded」を指定してください。

発行されたトークンは、有効期限が経過するか、一度 API で利用されると、無効となります。

複数の API でトークンを利用される場合は、tokenNumber にてトークンを複数発行してください。

No	パラメータ名	必須	型	桁	意味	備考
1	Encrypted	◎	CHAR	-	カード情報を暗号化した情報	カード番号等の情報をEncrypted作成手順に基づき、作成した値を設定してください。※1
2	ShopID	◎	CHAR	13	ショップID	弊社が発行する値を設定してください。
3	KeyHash	◎	CHAR	-	公開鍵のハッシュ値	管理画面からダウンロードした公開鍵のハッシュ値を設定してください。

【必須項目の記号　◎：必須　●：条件により必須　空白：省略可能　-：出力時】

カード情報

名称	必須	意味	設定する値
cardNo	◎	トークン化するカード番号	カード番号を、ハイフンなし/半角数字で設定してください。
expire	◎	トークン化するカードの有効期限	カード有効期限を、半角数字で YYMM 又は YYYYMM 形式いずれかで設定してください。
securityCode		トークン化するカードのセキュリティコード	セキュリティコードを､3 又は 4 桁の数字で設定してください。設定されない場合、セキュリティコードを利用せずに決済を行います。
holderName		トークン化するカードのカード名義人	カード名義人を、以下の文字種で 50 文字以内で設定してください。設定した場合は、取引詳細に出力されます。 文字種：(全て半角) 数字/アルファベット(大文字/小文字)/ “ “（スペース）,(コンマ）.(ピリオド)-(ハイフン)/(スラッシュ)
tokenNumber		トークンを発行する数	発行するトークンの数を 1～10 の数値で設定してください。 入力値無しの場合は､1 個のトークンを発行します。

【必須項目の記号　◎：必須　●：条件により必須　空白：省略可能　-：出力時】

※1 Encrypted 作成手順
①カード情報は json 形式で構築します。
例：
{"cardNo":"4111111111111111","expire":"2001","securityCode":"123","holderName":"TAROYAMADA","tokenNumber":"2"}
②①を PG マルチペイメントサービスから提供された公開鍵を使って RSA 方式(2048bit)で暗号化します。
③BASE64 でエンコードします。

4.2.4. 返却パラメーター(当サービス⇒スマートフォンアプリ)

Json 形式で結合したパラメータを出力します。

また、HTTP レスポンスの Content-Type ヘッダの MIME タイプは「application/json」にて出力します。

名称		意味	設定する値
resultCode		処理結果を表すコード	別表 5.3.の結果コードが配列で設定されます。
tokenObject		取得したトークン情報 以下のプロパティを持つオブジェクト	トークン情報が、object で設定されます。 なお、resultCode が"000"(取得完了)以外の場合、このオブジェクト自身もしくはプロパティが設定されない場合があります。
	token	トークン配列	決済時には、この値を設定してください。トークン文字列を配列にした値が設定されます。
	toBeExpiredAt	トークン有効期限	トークンの有効期限です。 yyyy-mm-dd-HH-MM-SS 形式で設定されます。
	maskedCardNo	マスク済みカード番号	トークン化されたカード番号が、一部*マスクされて設定されます。
	isSecurityCodeSet	セキュリティコード設定フラグ	トークン取得時にセキュリティコードが設定された事を表すフラグです。bool 値で設定され、true の場合セキュリティコード設定ありとされます。

※返却パラメーターサンプル

・tokennumber 設定なしの場合

{"tokenObject":{"token":["a33c8bec609ffc75726249d8d82286d529bd1deb973119cf497eeb54610ab9d2"],"toBeExpiredAt":"2016-09-26-17-56-38","maskedCardNo":"411111*******111","isSecurityCodeSet":true},"resultCode":["000"]}

・tokennumber=3 の場合

{"tokenObject":{"token":["a33c8bec609ffc75726249d8d82286d529bd1deb973119cf497eeb54610ab9d2","c89d7a6484af2b116e9a8636f8c1020de885bf2765a3ab3565f20f029dcc3150","719bfb8ec852c0b56a4cde92336e7f0a142a8e25f9018f3f1ec129817b87bc98"],"toBeExpiredAt":"2016-09-26-17-56-38","maskedCardNo":"411111*******111","isSecurityCodeSet":true},"resultCode":["000"]}

・エラーの場合

{"tokenObject":{"token":[""],"toBeExpiredAt":"","maskedCardNo":"","isSecurityCodeSet":""},"resultCode":["551"]}

{"tokenObject":{"token":[""],"toBeExpiredAt":"","maskedCardNo":"","isSecurityCodeSet":""},"resultCode":["551","550"]}

4.2.5. サンプルコード

4.2.5.1 Android (Java 版)

```
import java.security.KeyFactory;
import java.security.PublicKey;
import java.security.spec.X509EncodedKeySpec;
import javax.crypto.Cipher;
import org.apache.commons.codec.binary.Base64;
import org.json.JSONObject;

// 中略

    // PG の公開鍵
    private static final String PG_PUBLIC_KEY = "....";

    //
    // Encrypted 作成手順例
    // (例外処理については別途考慮してください)
    //
    public String makeEncryptedCardInfo() throws Exception {

        // カード情報 JSON データ作成
        Map<String, String> map = new HashMap<>();
        map.put("cardNo", xxxxxxxxxxxxxxx);  // カード番号
        map.put("expire", "201907");   // カードの有効期限
        map.put("securityCode", xxx);  // セキュリティ コード(任意項目)
        map.put("holderName", xxxx);  // カード名義人(任意項目)
        map.put("tokenNumber", 1);  // トークンを発行する数

        String cardInfoJson = new JSONObject(map).toString();

        // PG マルチペイメントサービスから提供された公開鍵から公開鍵オブジェクト作成
        byte[] buffer = Base64.decodeBase64(PG_PUBLIC_KEY.getBytes());
        X509EncodedKeySpec keySpec = new X509EncodedKeySpec(buffer);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        PublicKey publicKey = keyFactory.generatePublic(keySpec);

        // カード情報 JSON データを暗号化
        Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        byte[] encryptedByte = cipher.doFinal(cardInfoJson.getBytes());
```

```
        // Base64 エンコード処理
        String encryptedCardInfo = Base64.encodeToString(encryptedByte, Base64.DEFAULT);

        return encryptedCardInfo;
    }
```

4.2.5.2 iOS (Swift 版)

```
import UIKit
import Foundation

// 中略

    //
    // カード情報を暗号化した情報(Encrypted)作成メソッド
    //
    func makeCardInfoData() -> String? {

        // カード情報 JSON データ作成
        var jsonObj = Dictionary<String,Any>()
        jsonObj["cardNo"] = xxxxxxxxxxxxxxx  // カード番号
        jsonObj["expire"] = "201907"  // カードの有効期限
        jsonObj["securityCode"] = xxx  // セキュリティ コード(任意項目)
        jsonObj["holderName"] = xxxx  // カード名義人(任意項目)
        jsonObj["tokenNumber"] = 1  // トークンを発行する数

        do{
            let jsonData = try JSONSerialization.data(withJSONObject: jsonObj, options: [])
            let cardInfo = String(bytes: jsonData, encoding: .utf8)!

            // PG マルチペイメントサービスから提供された公開鍵から公開鍵オブジェクト作成
            let gmoPubKeyStr = "xxxxxxxx";
            let data:Data = Data(base64Encoded: gmoPubKeyStr, options:
                    Data.Base64DecodingOptions.ignoreUnknownCharacters)!
            let keyDict:[NSObject:NSObject] = [
                kSecAttrKeyType: kSecAttrKeyTypeRSA,
                kSecAttrKeyClass: kSecAttrKeyClassPublic,
                kSecAttrKeySizeInBits: NSNumber(value: 2048),
                kSecReturnPersistentRef: true as NSObject
            ]
            let publicKey = SecKeyCreateWithData(data as CFData, keyDict as CFDictionary, nil)

            // カード情報 JSON データを暗号化
            let plainBuffer = [UInt8](cardInfo.utf8)
            var cipherBufferSize : Int = Int(SecKeyGetBlockSize(publicKey!))
            var cipherBuffer = [UInt8](repeating:0, count:Int(cipherBufferSize))
```

```
            let status = SecKeyEncrypt(publicKey!, SecPadding.PKCS1, plainBuffer, plainBuffer.count,
                    &cipherBuffer, &cipherBufferSize)
            if (status != errSecSuccess) {
                // 失敗時の処理
            }

            // Base64 エンコード処理
            let mudata = NSData(bytes: &cipherBuffer, length: cipherBufferSize)
            let encryption = mudata.base64EncodedString(options:
                    NSData.Base64EncodingOptions.lineLength64Characters)
            return encryption
        } catch {
            return nil
        }
    }
```

4.3. API 版注意事項

4.3.1. 公開鍵について

公開鍵は管理画面からダウンロードしてください。

ショップ管理画面 > 都度決済 > クレジットカード > 設定 > トークン API/キー情報

※テスト用と本番用の公開鍵は異なります。

4.3.2. 対応アプリ通信方式

TLS は 1.2 に対応しています。

ATS につきましては対応しておりませんので、アプリ側で ATS に対応している場合は、ホワイトリストへの URL の登録をお願いします。

検証環境：https://pt01.mul-pay.jp/ext/api/credit/getToken

本番環境：https://p01.mul-pay.jp/ext/api/credit/getToken

5. その他注意事項

5.1. トークン受信後のサーバー処理

5.1.1. トークンを利用してマルチペイメントサービスの API を実行する

購入フォームを受信した加盟店様サーバー処理にて、マルチペイメントの API を実行します。

詳細な API の仕様は、以下仕様書をご覧ください。

プロトコルタイプ(マルチ決済_インタフェース仕様)

モジュールタイプ(Java 版_マルチ決済インタフェース仕様)

モジュールタイプ(PHP 版_マルチ決済インタフェース仕様)

カード決済インタフェース仕様書はドキュメント統合しましたので、今後は Web ドキュメントからご参照ください。

・上記各仕様書における、以下のカード情報に関する項目を、受信したトークンに置き換えてください。

置き換え対象項目

（クレジットカード決済、多通貨クレジットカード決済（DCC）の場合）

No	パラメータ名
1	CardNo
2	Expire
3	SecurityCode

置き換え対象項目（カード登録／更新の場合）

No	パラメータ名
1	CardNo
2	Expire
3	HolderName
4	SecurityCode

置き換え後項目

No	パラメータ名
1	Token

・その他の項目については、各 API の仕様に従います。

5.2. トークンの有効期限について

発行されたトークンは、有効期限が経過するか、一度 API で利用されると、無効となります。

有効期限はトークンが発行されてから約 30 分となります。

有効期限は予告なしに変更される場合があります。

5.3. 処理結果コード表

トークン取得結果で発生するエラーコードを、以下に記述します。000 番が正常終了となり、トークンが発行されている状態です。500 番以下のエラーは、パラメータ不正となります。コードに従って、エンドユーザーに再入力を促し、リトライ可能です。500 より大きなエラーは、原則サーバー側もしくは当社設定の異常となりますので、お問い合わせください。

コード	意味
000	トークン取得正常終了
100	カード番号必須チェックエラー
101	カード番号フォーマットエラー(数字以外を含む)
102	カード番号フォーマットエラー(10-16 桁の範囲外)
110	有効期限必須チェックエラー
111	有効期限フォーマットエラー(数字以外を含む)
112	有効期限フォーマットエラー(6 又は 4 桁以外)
113	有効期限フォーマットエラー(月が 13 以上)
121	セキュリティコードフォーマットエラー(数字以外を含む)
122	セキュリティコード桁数エラー
131	名義人フォーマットエラー(半角英数字、一部の記号以外を含む)
132	名義人フォーマットエラー(51 桁以上)
141	発行数フォーマットエラー(数字以外を含む)
142	発行数フォーマットエラー(1-10 の範囲外)
150	カード情報を暗号化した情報必須チェックエラー
160	ショップ ID 必須チェックエラー
161	ショップ ID フォーマットエラー(14 桁以上)
162	ショップ ID フォーマットエラー(半角英数字以外)
170	公開鍵ハッシュ値必須チェックエラー
180	ショップ ID または公開鍵ハッシュ値がマスターに存在しない
190	カード情報(Encrypted)が復号できない
191	カード情報(Encrypted)復号化後フォーマットエラー
200	callback 必須チェックエラー
201	Callback フォーマットエラー(半角英数字、アンダースコア、ピリオド以外を含む)
501	トークン用パラメータ(id)が送信されていない
502	トークン用パラメータ(id)がマスターに存在しない
511	トークン用パラメータ(cardInfo)が送信されていない
512	トークン用パラメータ(cardInfo)が復号できない
521	トークン用パラメータ(key)が送信されていない
522	トークン用パラメータ(key)が復号できない
531	トークン用パラメータ(callBack)が送信されていない
541	トークン用パラメータ(hash)が存在しない
551	トークン用 apikey が存在しない ID
552	トークン用 apikey が有効ではない
701	トークン用パラメータ(cardObject)が存在しない
901	マルチペイメント内部のシステムエラー

902	処理が混み合っている