外部IF仕様説明書
（トークン決済）

第1.1.4版 2019年 3月 20日

株式会社ペイジェント

目次

1.はじめに

本書は、株式会社ペイジェント(以下、「ペイジェント」といいます）が提供する、トークン決済の仕様について説明しています。

本書に記載するコードサンプルについて、一切の責任を負いません。加盟店様にて十分な検証を行って頂きますようお願い致します。

2.トークン決済の概要

トークン決済とは、購入者が入力したカード情報をJavaScriptで別の文字列(トークン)に置き換え、そのトークンを使用してクレジットカード決済を行う仕組みです。

加盟店様はトークン決済を利用することで、ECサイトにクレジットカード情報を保持する必要がなくなります。

■カード情報のトークンを取得する

ペイジェントが提供するカード情報トークンJavaScriptへのリンクを加盟店様ECサイトに組み込みます。

購入者が入力したカード情報とトークン生成鍵をパラメータに設定して、カード情報トークンスクリプトを実行するとトークンが応答されます。

トークンへの置き換え対象となるカード情報の違いにより、トークンは2種類に大別されます。

クレジットカードトークン

購入者が新規にクレジットカード情報を入力してお支払いをする、もしくはクレジットカード情報を登録する際に使用します。トークンへ置き換える情報は以下のとおりです。

クレジットカード番号
有効期限
クレジットカード名義
セキュリティコード

セキュリティコードトークン

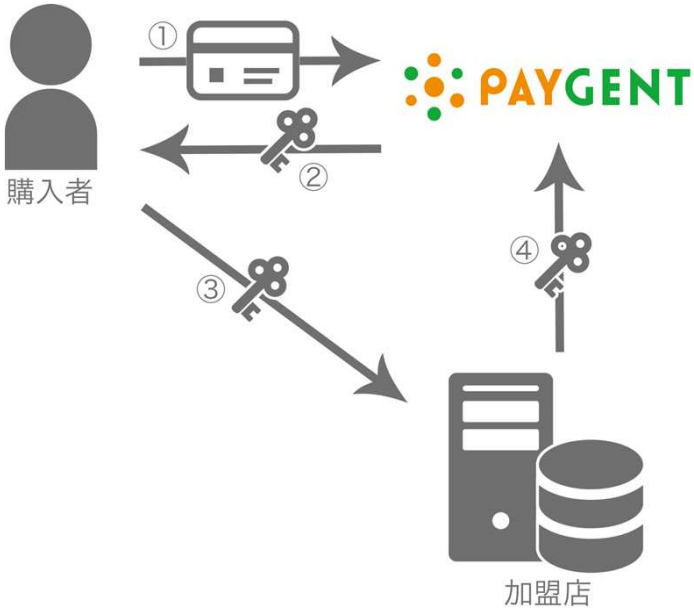
予め登録したクレジットカード情報を利用して、購入者がセキュリティコードだけをお支払のたびに入力する際に使用します。トークンへ置き換える情報は以下のとおりです。

セキュリティコード

■トークンを電文に設定する

JavaScriptによって応答されたトークンを、カード情報(カード番号、有効期限 etc）の代替としてカード決済オーソリ電文やカード情報登録電文に設定します。

<図 2.1トークン決済イメージ>

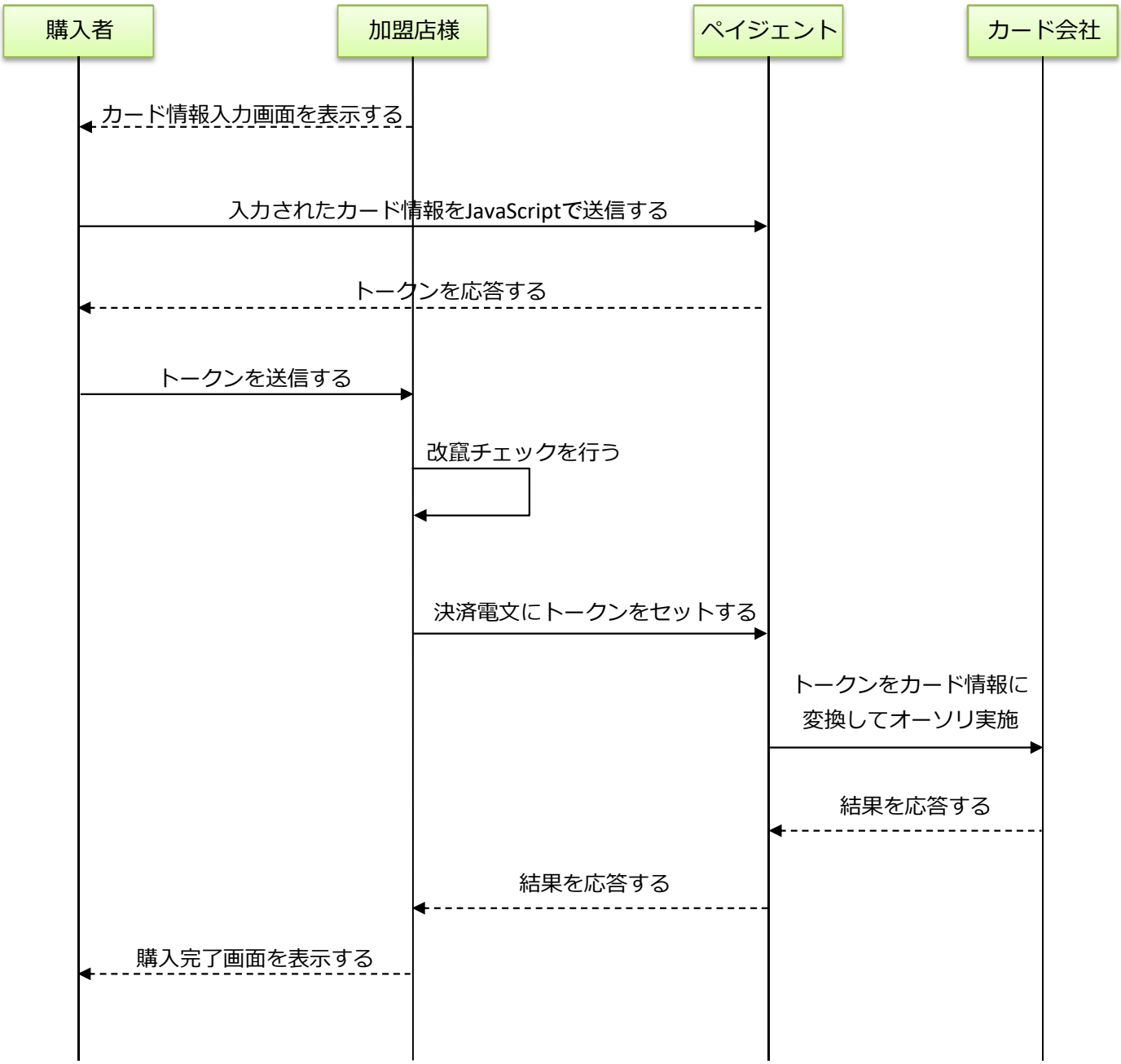


①JavaScript により、カード情報をペイジェントへ送信
②トークン返却
③トークンを加盟店サーバへ送信
④トークンでオーソリ実行

3.トークン決済の流れ

トークンを使ったクレジットカード決済の流れを以下に示します。

<図 3.1トークン決済の流れ>



4.トークンを取得する

4.1.トークンJavaScriptを読み込む

加盟店様サイトで、ペイジェントが提供するトークンJavaScriptを読み込んで下さい。

```
<!-- 文字コード UTF-8を指定して、PaygentToken.js を画面に組み込む。(環境に応じて要求先URLを変更する) -->
<script type="text/javascript" src="https://xxx/PaygentToken.js" charset="UTF-8"></script>
```

JavaScriptを読み込むURLは、試験環境と本番環境で異なります。

試験環境	https://sandbox.paygent.co.jp/js/PaygentToken.js
本番環境	https://token.paygent.co.jp/js/PaygentToken.js

購入者が入力したクレジットカード情報をペイジェントへ送信し、トークンを取得します。
作成するトークンの種類により利用するJavaScript関数が異なります。

トークンの種類	JavaScript関数名	処理内容
クレジットカードトークン	createToken()	カード番号、有効期限、カード名義、セキュリティコードを引数に設定して、トークン文字列を取得します。
セキュリティコードトークン	createCvcToken()	セキュリティコードを引数に設定して、トークン文字列を取得します。 予めペイジェントに登録したクレジットカード情報と組み合わせてご利用下さい。

4.2.1.クレジットカードトークンの場合

購入者が入力したクレジットカード情報をペイジェントへ送信し、トークンを取得します。

■createToken()のインターフェース

引数	名称	プロパティ名	必須/任意	説明
第１引数	マーチャントID		必須	ペイジェントよりお知らせしたマーチャントIDを設定して下さい。 マーチャントIDは試験環境と本番環境で異なります。
第２引数	トークン生成鍵		必須	ペイジェントオンライン(PGOL)にて確認した値を設定して下さい。 トークン生成鍵は試験環境と本番環境で異なります。
第３引数	カード情報	card_number	必須	カード番号。半角数字。14-16バイト。
		expire_year	必須	有効期限（年)。YY。 1桁で指定された場合、先頭1桁を0で補完します。
		expire_month	必須	有効期限（月)。MM。 1桁で指定された場合、先頭1桁を0で補完します。
		cvc	任意	セキュリティコード。最大4バイト。半角数字。
		name	任意	カード名義。最大64バイト。半角英数字、半角カナ、半角スペース。
第４引数	トークン後実行関数		必須	トークン化した後に実行するJavaScript関数名。

```
<script type="text/javascript">
<!-- send関数の定義。カード情報入力フォームの送信ボタン押下時の処理。-->
function send() {
    var form = document.card_form;
    var paygentToken = new PaygentToken();          //PaygentTokenオブジェクトの生成
    paygentToken.createToken(
        '10001',                                    //第１引数：マーチャントID
        'live_00000000000000000000000000',          //第２引数：トークン生成鍵
        {                                           //第３引数：クレジットカード情報
            card_number:form.card_number.value,         //クレジットカード番号
            expire_year:form.expire_year.value,         //有効期限-YY
            expire_month: form.expire_month.value,      //有効期限-MM
            cvc:form.cvc.value,                         //セキュリティーコード
            name:form.name.value                        //カード名義
        },execPurchase                              //第４引数：コールバック関数(トークン取得後に実行)
    );
}
</script>
```

4.2.2.セキュリティコードトークンの場合

購入者が入力したセキュリティコードをペイジェントへ送信し、トークンを取得します。

■createCvcToken()のインターフェース

引数	名称	プロパティ名	必須/任意	説明
第１引数	マーチャントID		必須	ペイジェントよりお知らせしたマーチャントIDを設定して下さい。 マーチャントIDは試験環境と本番環境で異なります。
第２引数	トークン生成鍵		必須	ペイジェントオンライン(PGOL)にて確認した値を設定して下さい。 トークン生成鍵は試験環境と本番環境で異なります。
第３引数	カード情報	cvc	必須	セキュリティコード。最大4バイト。半角数字。
第４引数	トークン後実行関数		必須	トークン化した後に実行するJavaScript関数名。

```
<script type="text/javascript">
<!-- send関数の定義。カード情報入力フォームの送信ボタン押下時の処理。-->
function send() {
    var form = document.card_form;
    var paygentToken = new PaygentToken();          //PaygentTokenオブジェクトの生成
    paygentToken.createCvcToken(
        '10001',                                    //第１引数：マーチャントID
        'live_00000000000000000000000000',          //第２引数：トークン生成鍵
        {                                           //第３引数：クレジットカード情報
            cvc:form.cvc.value,                         //セキュリティーコード
        },execPurchase                              //第４引数：コールバック関数(トークン取得後に実行)
    );
}
</script>
```

4.3.1.クレジットカードトークンの場合

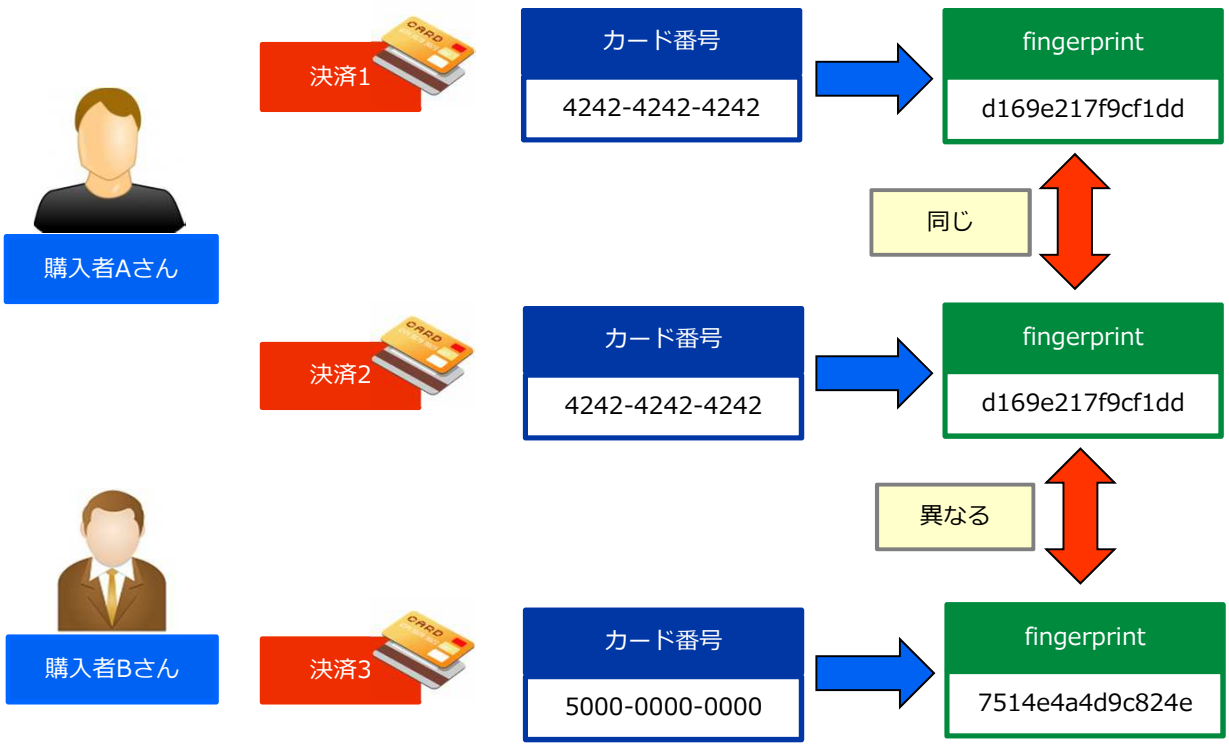
加盟店様が指定したトークン後実行関数において、取得したトークンを加盟店様サイトへ送信してください。
createToken()の処理後には、下記のJavaScriptオブジェクトが応答されます。

名称	プロパティ名	説明
処理結果コード	result	「付録A - 処理結果コード一覧」参照。
トークン化カードオブジェクト	tokenizedCardObject	トークン化されたカード情報をJavaScript Objectで設定します。 処理結果コードが正常(0000)以外は設定されません。
トークン	token	トークン文字列。
マスクされたカード番号	masked_card_number	カード番号の下4桁以外をマスク化します。 **********4242。
トークン有効期限	valid_until	トークンの有効期限。YYYYMMDDHHmmss。 トークンは一度使われるか、この有効期限を超過すると無効になります。
フィンガープリント	fingerprint	カード番号を一意に識別できる値。 ※fingerprintの書式は最大64バイトの英数字です。
チェックハッシュ	hc	改竄チェックに用いるハッシュ値。 チェック方法は「5.改竄チェックを行う 」をご参照ください。

フィンガープリント（fingerprint）について

fingerprintとはカード番号の代わりにカード番号を一意に識別できる値です。
同じカード番号であれば、同じfingerprintを応答します。
加盟店様(マーチャントID)ごとに、fingerprintを生成します。

<fingerprintイメージ>



```
<!-- コールバック関数の定義。トークン取得後の処理-->
 function execPurchase(response) {
    var form = document.card_form;
    if (response.result == '0000') {                    //トークン処理結果が正常の場合
        <!-- カード情報入力フォームから、入力情報を削除。-->
        form.card_number.removeAttribute('name');
        form.expire_year.removeAttribute('name');
        form.expire_month.removeAttribute('name');
        form.cvc.removeAttribute('name');
        form.name.removeAttribute('name');

        <!-- 予め用意したhidden項目にcreateToken()から応答されたトークン等を設定。-->
        form.token.value = response.tokenizedCardObject.token;
        form.masked_card_number.value = response.tokenizedCardObject.masked_card_number;
        form.valid_until.value = response.tokenizedCardObject.valid_until;
        form.fingerprint.value = response.tokenizedCardObject.fingerprint;
        form.hc.value = response.hc;
        <!-- カード情報入力フォームをsubmitしてtokenを送信する -->
        form.submit();

    } else {                                            //トークン処理結果が異常の場合
        <!-- エラー時の処理をここに記述する -->
    }
}
</script>
```

トークンを加盟店様サイトに送信する前に、入力されたカード情報を削除して下さい。
削除しない場合、加盟店様サイトの実装によっては、購入者が入力したカード情報が加盟店様サイトに送信される可能性があります。

加盟店様が指定したトークン後実行関数において、取得したトークンを加盟店様サイトへ送信してください。
createCvcToken()の処理後には、下記のJavaScriptオブジェクトが応答されます。

名称	プロパティ名	説明
処理結果コード	result	「付録A - 処理結果コード一覧」参照。
トークン化カードオブジェクト	tokenizedCardObject	トークン化されたカード情報をJavaScript Objectで設定します。 処理結果コードが正常(0000)以外は設定されません。
トークン	token	トークン文字列。
トークン有効期限	valid_until	トークンの有効期限。YYYYMMDDHHmmss。 トークンは一度使われるか、この有効期限を超過すると無効になります。

```
<!-- コールバック関数の定義。トークン取得後の処理-->
 function execPurchase(response) {
    var form = document.card_form;
    if (response.result == '0000') {                    //トークン処理結果が正常の場合
        <!-- カード情報入力フォームから、入力情報を削除。-->
        form.cvc.removeAttribute('name');

        <!-- 予め用意したhidden項目tokenにcreateCvcToken()から応答されたトークンを設定。-->
        form.token.value = response.tokenizedCardObject.token;
        <!-- カード情報入力フォームをsubmitしてtokenを送信する -->
        form.submit();

    } else {                                            //トークン処理結果が異常の場合
        <!-- エラー時の処理をここに記述する -->
    }
}
</script>
```

トークンを加盟店様サイトに送信する前に、入力されたセキュリティコードをを削除して下さい。
削除しない場合、加盟店様サイトの実装によっては、購入者が入力したセキュリティコードが加盟店様サイトに送信される可能性があります。

購入フォーム全体のイメージを下記に示します。

4.4.1.クレジットカードトークンの場合

```
<!DOCTYPE html>
<html lang="ja">
<head>
 <!-- 文字コード UTF-8を指定して、Paygent.js を画面に組み込む。(環境に応じて要求先URLを変更する) -->
 <script type="text/javascript" src="http://xxx/PaygentToken.js" charset="UTF-8"></script>
 <script type="text/javascript">

 <!-- send関数の定義。カード情報入力フォームの送信ボタン押下時の処理。-->
  function send() {
   var form = document.card_form;
   var paygentToken = new PaygentToken();      //PaygentTokenオブジェクトの生成

  <!-- トークン生成メソッドの実行。-->
   paygentToken.createToken(
    '10001',                                    //第１引数：マーチャントID
    'live_00000000000000000000000000',          //第２引数：トークン生成鍵
    {                                           //第３引数：クレジットカード情報
     card_number:form.card_number.value,        //クレジットカード番号
     expire_year:form.expire_year.value,        //有効期限-YY
     expire_month: form.expire_month.value,     //有効期限-MM
     cvc:form.cvc.value,                        //セキュリティーコード
     name:form.name.value                       //カード名義
    },execPurchase                              //第４引数：コールバック関数(トークン取得後に実行)
   );
  }

 <!-- コールバック関数の定義。トークン取得後の処理-->
  function execPurchase(response) {
   var form = document.card_form;
   if (response.result == '0000') {             //トークン処理結果が正常の場合
      <!-- カード情報入力フォームから、入力情報を削除。（入力されたカード情報を、送信しないようにする。）-->
       form.card_number.removeAttribute('name');
       form.expire_year.removeAttribute('name');
       form.expire_month.removeAttribute('name');
       form.cvc.removeAttribute('name');
       form.name.removeAttribute('name');

      <!-- 予め用意したhidden項目にcreateToken()から応答されたトークン等を設定。-->
       form.token.value = response.tokenizedCardObject.token;
       form.masked_card_number.value = response.tokenizedCardObject.masked_card_number;
       form.valid_until.value = response.tokenizedCardObject.valid_until;
       form.fingerprint.value = response.tokenizedCardObject.fingerprint;
       form.hc.value = response.hc;

      <!-- カード情報入力フォームを submit。-->
       form.submit();
```

→次ページへ続く

```
    } else {                                        //トークン処理結果が異常の場合
      <!-- エラー時の処理をここに記述する -->
    }
  }
 </script>
</head>

<body>
<!-- カード情報入力フォーム -->
 <form action="xxxxxxxxx(遷移先URL)" method="POST" name="card_form">
  <div>カード番号：<input type="text" name="card_number"></div>
  <div>有効期限（年）：<input type="text" name="expire_year"></div>
  <div>有効期限（月）：<input type="text" name="expire_month"></div>
  <div>セキュリティコード：<input type="text" name="cvc"></div>
  <div>カード名義：<input type="text" name="name"></div>
  <input type="button" name="btn" value="送信" onClick="send();">
  <!-- 取得したトークン等をセットするhidden項目 -->
  <input type="hidden" name="token" value="">
  <input type="hidden" name="masked_card_number" value="">
  <input type="hidden" name="valid_until" value="">
  <input type="hidden" name="fingerprint" value="">
  <input type="hidden" name="hc" value="">
 </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
 <!-- 文字コード UTF-8を指定して、Paygent.js を画面に組み込む。(環境に応じて要求先URLを変更する) -->
 <script type="text/javascript" src="http://xxx/PaygentToken.js" charset="UTF-8"></script>
 <script type="text/javascript">

 <!-- send関数の定義。カード情報入力フォームの送信ボタン押下時の処理。-->
  function send() {
    var form = document.card_form;
    var paygentToken = new PaygentToken();      //PaygentTokenオブジェクトの生成

   <!-- トークン生成メソッドの実行。-->
    paygentToken.createCvcToken(
      '10001',                                  //第１引数：マーチャントID
      'live_00000000000000000000000000',        //第２引数：トークン生成鍵
      {                                         //第３引数：クレジットカード情報
        cvc:form.cvc.value,                     //セキュリティーコード
      },execPurchase                            //第４引数：コールバック関数(トークン取得後に実行)
    );
  }

 <!-- コールバック関数の定義。トークン取得後の処理-->
  function execPurchase(response) {
    var form = document.card_form;
    if (response.result == '0000') {            //トークン処理結果が正常の場合
        <!-- カード情報入力フォームから、入力情報を削除。（入力されたカード情報を、送信しないようにする。）-->
         form.cvc.removeAttribute('name');

        <!-- 予め用意したhidden項目tokenにcreateToken()から応答されたトークンを設定。-->
         form.token.value = response.tokenizedCardObject.token;

        <!-- カード情報入力フォームを submit。-->
         form.submit();
     } else {                                   //トークン処理結果が異常の場合
       <!-- エラー時の処理をここに記述する -->
     }
  }
 </script>
</head>
```

→次ページへ続く

```
<body>
<!-- カード情報入力フォーム -->
  <form action="xxxxxxxxx(遷移先URL)" method="POST" name="card_form">
    <div>セキュリティコード：<input type="text" name="cvc"></div>
    <input type="button" name="btn" value="送信" onClick="send();">
    <!-- 取得したトークンをセットするhidden項目 -->
    <input type="hidden" name="token" value="">
  </form>
</body>
</html>
```

5.改竄チェックを行う

以下の手順で、加盟店様のサーバ側でハッシュ値が一致するか確認することで、改竄リスクを回避出来ます。

決してJavaScript上でチェックを行わないようにしてください。
JavaScriptのチェック内容を解析されてしまうことで、ハッシュ値自体を改竄されることになります。

※改竄チェックを行わなくてもトークン自体は安全です。付加項目の改竄リスクの回避にご利用ください。

■チェックに使用するもの

ハッシュ値生成アルゴリズム	SHA-256アルゴリズム
ハッシュ値生成に使用する応答項目(※1)	トークン
	マスクされたカード番号(※2)
	トークン有効期限
	フィンガープリント
ハッシュ値生成に使用する鍵	ペイジェントオンラインで確認した「トークン受取ハッシュ鍵」 ※ハッシュ鍵は加盟店様ごとに異なります。 ペイジェントが提供する加盟店管理者サイトにログインして、 ハッシュ鍵を確認してください。
比較対象の応答項目(※1)	チェックハッシュ

※1 応答項目：createToken()処理後に応答されるJavaScriptオブジェクト(前述)
※2 マスクされたカード番号は"*"も含めたすべてを使用します。

■チェック手順

1. ハッシュ値生成に使用する応答項目と、ハッシュ値生成に使用する鍵を順に連結します。
2. 連結した文字列を、ハッシュ値生成アルゴリズムを用いてハッシュ値を生成します。
3. 生成したハッシュ値と、比較対象の応答項目の値が一致した場合は正常です。

■チェック手順の例

トークン(応答項目)	tok_aaaabbbbccccdddd
マスクされたカード番号(応答項目)	**********4242
トークン有効期限(応答項目)	20181121185202
フィンガープリント(応答項目)	A1b2C3d4E5f6G7h8I9j0K
ハッシュ値生成に使用する鍵	ch_eeeeffffgggghhhhiiii

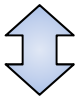
※上から順に文字列を連結します。

1. 文字列を連結

tok_aaaabbbbccccdddd**********4242201811211851 85202A1b2C3d4E5f6G7h8I9j0Kch_eeeeffffgggghhhhiiii

2. 連結した文字列をハッシュ化（SHA-256）

4d3c9a7b3f8dd69787e35491f0fa9e750b9642413ad6bc2100af9bba4edaaad9



一致：OK

3. ハッシュ値を比較

チェックハッシュ(応答項目)
4d3c9a7b3f8dd69787e35491f0fa9e750b9642413ad6bc2100af9bba4edaaad9

6.クレジットカード決済を行う

6.1.クレジットカードトークンを決済電文に設定する

クレジットカードトークンをカード情報(カード番号、有効期限、セキュリティコード、名義)の代わりに各種電文に設定します。

クレジットカードトークンに対応している電文を下記に示します。

<表 6.1クレジットカードトークンを設定できる電文一覧>

名称	電文種別ID	クレジットカードトークンに置き換えられる項目	
		名称	ID
カード決済オーソリ電文	020	カード番号	card_number
		カード有効期限	card_valid_term
		カード確認番号	card_conf_number
カード決済(多通貨)オーソリ電文	180	カード番号	card_number
		カード有効期限	card_valid_term
		カード確認番号	card_conf_number
カード情報設定電文	025	カード番号	card_number
		カード有効期限	card_valid_term
		カード名義人	cardholder_name

電文にクレジットカードトークンを設定した場合は、他の方法によるカード情報の指定はできません。

クレジットカードトークンと併用できないカード情報指定方法を下記に示します。

<表 6.2クレジットカードトークンと併用できないカード情報指定方法>

カード情報指定方法	説明
カード情報を直接指定する	カード番号、有効期限、確認番号を直接指定する方法。
過去に作成したカード決済情報を指定する	参照マーチャント取引IDを指定して、過去に作成したカード決済のカード情報を利用する方法。
お預かりカード情報を指定する	顧客ID、顧客カードIDを指定して、ペイジェントに登録したカード情報を利用する方法。

「カード決済オーソリ電文」、「カード決済(多通貨)オーソリ電文」でクレジットカードトークンを利用する場合、項目「セキュリティーコードトークン利用」には「利用しない(NULLもしくは0)」を設定してください。

6.2.セキュリティコードトークンを決済電文に設定する

セキュリティコードトークンをセキュリティコードの代わりに各種電文に設定します。
セキュリティコードトークンに対応している電文を下記に示します。

<表 6.3セキュリティコードトークンを設定できる電文一覧>

名称	電文種別ID	セキュリティコードトークンに置き換えられる項目	
		名称	ID
カード決済オーソリ電文	020	カード確認番号	card_conf_number
カード決済(多通貨)オーソリ電文	180	カード確認番号	card_conf_number

電文にセキュリティコードトークンを設定した場合は、下記に示すカード情報の指定が必要となります。

<表 6.4セキュリティコードトークンと併用で指定が必要な方法>

カード情報指定方法	説明
お預かりカード情報を指定する	顧客ID、顧客カードIDを指定して、ペイジェントに登録したカード情報を利用する方法。

電文にセキュリティコードトークンを設定した場合は、他の方法によるカード情報の指定はできません。
トークンと併用できないカード情報指定方法を下記に示します。

<表 6.5セキュリティコードトークンと併用できないカード情報指定方法>

カード情報指定方法	説明
カード情報を直接指定する	カード番号、有効期限、確認番号を直接指定する方法。
過去に作成したカード決済情報を指定する	参照マーチャント取引IDを指定して、過去に作成したカード決済のカード情報を利用する方法。

「カード決済オーソリ電文」、「カード決済(多通貨)オーソリ電文」でセキュリティコードトークンを利用する場合、項目「セキュリティーコードトークン利用」には「利用する(1)」を設定してください。

7.注意事項

7.1.トークンの有効期限

トークンは下記条件で無効になります。

・１回でも電文の応答が正常で返ってきた。
・トークン有効期限が切れた。
　→createToken()実行後に応答するtokenizedCardObject.valid_untilに有効期限が設定されています。

7.2.トークン生成鍵・トークン受取ハッシュ鍵

トークン生成鍵、トークン受取ハッシュ鍵はペイジェントオンラインで確認してください。
　※トップページ > システム情報管理
トークン生成鍵、トークン受取ハッシュ鍵を変更したい場合は、リセット機能をご利用ください。
リセットしますと変更前の鍵は無効となりますので、ご利用には十分ご注意ください。

<図 7.1 ペイジェントオンライン トークン生成鍵・トークン受取ハッシュ鍵確認画面>

7.3.サポートするブラウザ

トークン決済はJavaScriptに対応したブラウザの利用を前提としております。
JavaScriptが利用できない環境では正常に動作致しません。
2016年6月時点で下記ブラウザでの動作を確認しています。

環境	バージョン
Internet Explorer	9.0以降 ※バージョンが9.xの場合、httpで構築されたサイトからだとトークンJavaScriptが正常に動作しません。httpsでサイトを構築して下さい。
Google Chrome	49.0.26
Firefox	45.0.1
iOS(8.2) - Safari	600.1.4
Android(5.0.2)-標準ブラウザ	1.0
Android(5.0.2)-Chrome	51.0.2704.81

付録A - 処理結果コード一覧

トークン取得時に応答する処理結果コードの一覧を下記に示します。

処理結果コード	説明
0000	正常終了
1100	マーチャントID - 必須エラー
1200	トークン生成公開鍵 - 必須エラー
1201	トークン生成公開鍵　- 不正エラー
1300	カード番号 - 必須チェックエラー
1301	カード番号 - 書式チェックエラー
1400	有効期限(年) - 必須チェックエラー
1401	有効期限(年) - 書式チェックエラー
1500	有効期限(月) - 必須チェックエラー
1501	有効期限(月) - 書式チェックエラー
1502	有効期限(年月)が不正です。 年月として正しいかに加えて、以下の場合もエラーとします。 ・過去年月である ・未来20年以降である
1600	セキュリティコード - 書式チェックエラー
1601	セキュリティコード - 必須エラー（セキュリティコードトークンの場合）
1700	カード名義 - 書式チェックエラー
7000	非対応のブラウザです。
7001	ペイジェントとの通信に失敗しました。
8000	システムメンテナンス中です。
9000	ペイジェント決済システム内部エラー

改訂履歴

版数	日付	変更箇所	変更内容
1.0.0	2016年6月1日		初版
1.0.1	2016年7月27日	6.3	サポートするブラウザに動作を検証した環境を追記。
1.0.2	2016年11月9日	6.3	Internet Explorer9.0の場合の注意事項を追記。
1.1.0	2017年2月22日	全体	セキュリティコードトークンの仕様を追記。
1.1.1	2018年1月17日	4.3.1	クレジットカードトークンの応答項目にfingerprintを追記。
1.1.2	2018年9月19日	4.3.1 6.1	fingerprintの説明を修正。 トークンの有効期限の条件を修正
1.1.3	2018年11月21日	3 4.3.1 4.4.1 5 7.2	・図3.1に改竄チェックを追加。 ・トークン有効期限の書式表記をJavascriptのものに訂正。 ・クレジットカードトークンの応答項目にチェックハッシュを追加。 ・「5.改竄チェックを行う」を追加。 ・トークン受取ハッシュ鍵の記述を追加。
1.1.4	2019年3月20日	付録A	以下のエラーコードに対する説明を修正。 【エラーコード】 1301、1401、1501、1502、1600、1700