VeriTrans 4G

MDK Token Development Guide

Ver. 1.0.1 (April 2017~)

Table of Contents

Chapter 1 Getting Started with MDK Token

1-1 Contents of This Guide

This guide describes the information necessary for online stores to accept payment using VeriTrans 4G MDK Token provided by VeriTrans Inc.

1-2 Intended Audience

This guide is intended for developers of online stores who use the MDK Token. MDK Token can be used in conjunction with services listed in section "2-4 Payment Services That Supports MDK Token".

Title	Overview	Planner	Developer	Operator
MDK Token Development Guide	This guide describes an overview of the MDK Token and an API for getting the token.		◎	
VeriTrans4G Development Guide	This guide describes how to integrate VeriTrans 4G MDK (Merchant Development Kit).		◎	
VeriTrans4G Interface details ~Credit card payment~ ~MPI~	It describes payment interface specifications. Refer to this for API's which utilize MDK Token.		◎	

Table 1-2-1 List of documents

1-3 Limitation

The environment (browser) that consumers shall use must satisfy the following conditions:

- JavaScript should be enabled.
- It should support Cross-Origin Resource Sharing.
- It should support SHA-256 and should be compatible with TLS 1.1 or later.

* In the case of an environment which is not compatible with TLS 1.1 or higher, refer to the section "3.1.1 Access URL" and take the proper measures.

* Since feature phones do not support Cross-Origin Resource Sharing, the MDK Token cannot be used.

1-4 Copyrights and Contact Details

[Contact Details] Technical Support, VeriTrans Inc. E-mail: tech-support@veritrans.jp

1-5 Revision History

2017/2: Ver1.0.0 Release
2017/4: Ver1.0.1
Added description of "4-2 Sample code"

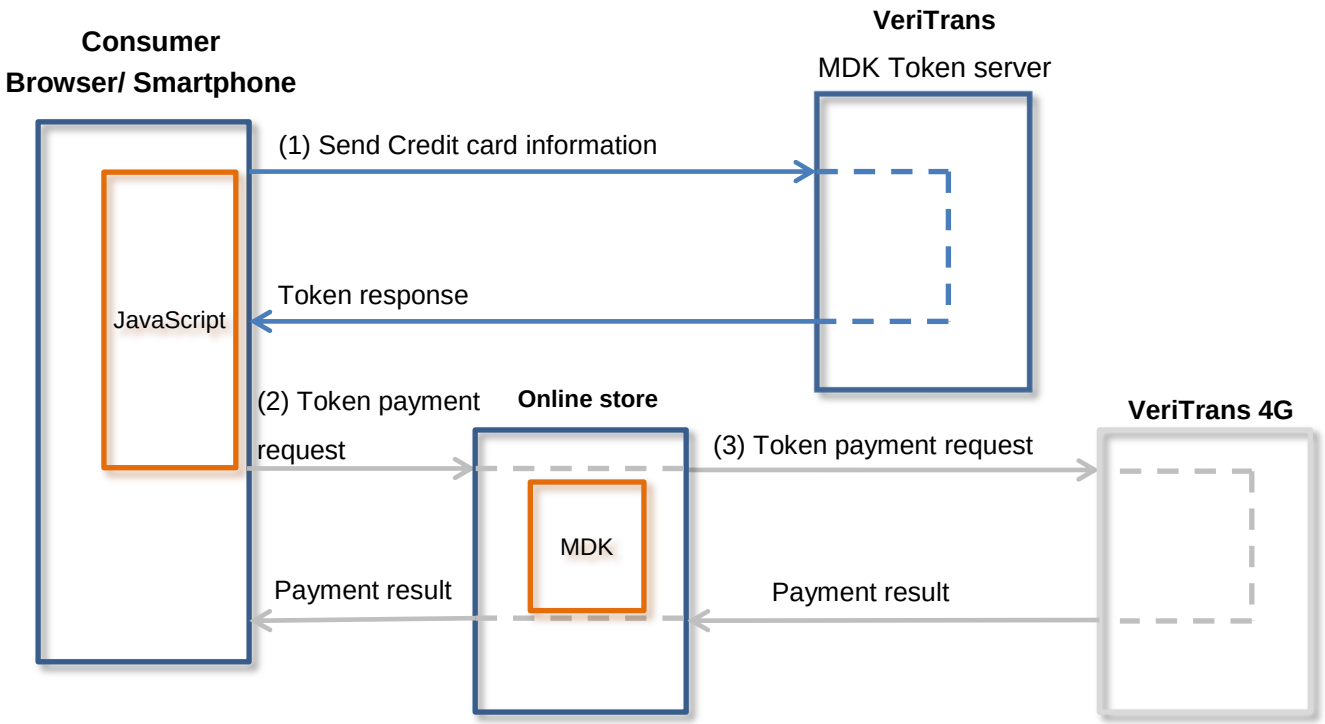
Chapter 2 Overview of MDK Token

2-1 Overview

- The MDK Token server (hereinafter, token server) issues the "token", necessary for implementation of the system which does not store or transmit card holder data.
- The credit card information is sent to the token server which responds with a token that can be used with VeriTrans 4G.
- At the time of acquiring a token, credit card information is sent directly from the consumer browser to the token server by using JavaScript. Payment can be processed by an online store without sending the credit card information. Online store will send the token instead of the credit card information in payment request to VeriTrans 4G.
- For the list of payment services that can use the token, refer to "2-4 Payment Services That Supports MDK Token ".

2-2 Payment Process Flow

Outline of payment process flow using token is as follows.



* Please refer to the development guide of each payment service for detailed process after step (2)

2-3 Precautions

- Tokens received from the token server can only be used once (one-time token).
- The token has an expiry date. New payments cannot be initiated using an expired token. For details, refer to the parameter "token_expire_date" in the response message in the section "3.2.1 Response message".
- When acquiring a token, use the token API key associated with the merchant ID specified at the time of payment.
  Credit card information cannot be matched if merchant ID differs at the time of token acquisition to that of payment.

2-4 Payment Services That Supports MDK Token

For details, Please refer to the development guide of each payment service which utilizes token.

Payment service name	Description
Credit card payment	This is a payment service using credit card. This service enables making payments with credit cards issued by various brands and card companies.
Credit card payment (with 3-D Secure)	This is a 3-D Secure authentication service. This service can be used as a standalone service or along with credit card payment for credit card brands which supports 3-D Secure.
Credit card payment (Member ID payment)	This is the service in which member and card information is registered at the same time as that of payment. Next time onwards payment can be processed only by specifying the member (without having to enter card information). It also enables consumers to manage card information (addition / deletion) by themselves. It is possible to skip card information input screen at the time of payment and charge the consumer with already registered credit card (Quick check out). * Payment with member ID is a function of Subscription Service (optional service). To use this service separate application is required.

Table 2-4-1 Payment service list

Chapter 3 Interface Details

This chapter explains about the message used at the time of acquiring the MDK Token.

Please refer the interface details given in a separate volume of development guide for each payment service.

3-1 Request Message

Specifications of request messages are as follows:

Protocol		HTTP1.1
Method		POST
Request header	**Accept**	application/json
	Content-Type	application/json
Port number		443
Character encoding		UTF-8
URL		Refer to “3.1.1Access URL”
Parameter		Refer to “3.1.2 Request message”

3.1.1 Access URL

Access URL (Official environment)	
Token server end point	https://api.veritrans.co.jp/4gtoken
* To access this end point a client that supports TLS 1.1 or higher is required	

Access URL (Temporary environment SSL 3.0 / TLS 1.0 valid)	
Token server end point	https://3gs.veritrans.co.jp/4gtoken
* This end point supports SSL 3.0 / TLS 1.0. * Use this end point if a client that does not support TLS 1.1 or higher. * This end point can be used until May 2018.	

* Since Android 4 series standard browser does not support TLS 1.1 or higher, please consider using a temporary environment as and when necessary.

3.1.2 Request Message Parameters

Below is a list of parameters that can be specified in the request message at the time of acquiring the MDK Token.

- The contents of the “Settings” column are as follows.
  - Required parameters: ○
  - Optional parameters: △

Request message: Consumer browser → Token server				
Field Name	**Item Name**	**Format/ Limitation**	**Explanation**	**Settings**
card_number	Card number	Single byte number; 19 digits or less	Numbers only	○
card_expire	Card expiry	Single byte number; symbol; 5 digits	MM/YY (Month +”/’ + year) format (For example, “08/18”)	○
security_code	Security code	Single byte number 3 or 4 digits	Security code	△
token_api_key	Token API key	Single byte alphanumeric; symbol; 36 digits or less	Key used for acquiring token	○
lang	Language	Refer to the right	Specify the expected language of the response message in request parameter "message" "en": English "ja": Japanese *If not specified then Accept-Language value will be used	△

3.1.3 Sample Request Message

```
{
  "card_number":"4111111111111111",
  "card_expire":"01/20",
  "security_code":"123",
  "token_api_key":"test-token-api-key",
  "lang":"en"
}
```

3-2 Response Message

Response message specifications are as follows.

Response Header	**Content-Type**	application/json
Character encoding		UTF-8
Parameter		Refer to “3.2.1 Response Message Parameter”

3.2.1 Response Message Parameter

The list of response message parameters from the token server is shown below.

- Contents of the “Settings” column are as follows.
  - Return always: ○
  - Return only at successful processing: Δ

Response message: Token server → Consumer browser				
Field name	**Item name**	**Format/ limitation**	**Explanation**	**Settings**
token	Token	Single byte alphanumeric; symbol; 36 digits or less	The value of the token used to identify the credit card information at the time of payment.	Δ
token_expire_date	Token expiry	Single byte number; 14 digits	Format of YYYYMMDDhhmmss (15 minutes after issue)	Δ
req_card_number	Request card number	Single byte number; symbol; 19 digits or less	Value set in the request message. Only first 6 and last 2 digits are displayed. Remaining characters are masked with "*" (asterisk). (Example "411111********11)	Δ
status	Status	Refer to the right	“success”: Success "failure": Failure	○
code	Result code	Single byte alphanumeric; symbol	Processing result code	○
message	Result message	String	Display processing result in English or in Japanese	○

* Store the value of the "token_expire_date" parameter, so that you can confirm whether the reason of failure to get credit card information at VeriTrans 4G side is because of token expiry.

*For result code/ result message list, refer to separate sheet “Token_ResultCodeList”.

3.2.2 Sample Response Message

•When token acquisition is successful.


```
{
  "token":"3a024e60-5951-4521-81e8-db0003752af8",
  "token_expire_date":"20161012182641",
  "req_card_number":"411111********11",
  "status":"success",
  "code":"success",
  "message":"Token has been successfully created."
}
```

•When token acquisition fails (format error in card number)

```
{
    "status": "failure",
    "code": "invalid_card_number",
    "message": "Card number is invalid."
}
```

•When token acquisition fails (merchant authorization error)

```
{
    "status": "failure",
    "code": "unauthorized_merchant",
    "message": " Merchant authorization error occurred."
}
```

Chapter 4 Other Supplementary Items

4-1 Testing

Please confirm whether MDK Token acquired from the token server can be used with each payment service.

For the test specification of each payment service and test card information refer to "VeriTrans 4G integration test guide".

[Important point]

When conducting a dummy transaction using test merchant ID, please use the token API key associated to the test merchant ID.

4-2 About Sample Code

Please refer the following JavaScript code sample for acquiring MDK token.

```
function submitToken(e) {
    var data = {};
    data.token_api_key = document.getElementById('token_api_key').innerText;
    if (document.getElementById('card_number')) {
        data.card_number = document.getElementById('card_number').value;
    }
    if (document.getElementById('cc-exp')) {
        data.card_expire = document.getElementById('cc-exp').value;
    }
    if (document.getElementById('cc-csc')) {
        data.security_code = document.getElementById('cc-csc').value;
    }
    data.lang = "ja";

    var url = document.getElementById('token_api_url').innerText;

    var xhr = new XMLHttpRequest();
    xhr.open('POST', url, true);
    xhr.setRequestHeader('Accept', 'application/json');
    xhr.setRequestHeader('Content-Type', 'application/json; charset=utf-8');
    xhr.addEventListener('loadend', function () {
        if (xhr.status === 0) {
            alert("トークンサーバーとの接続に失敗しました");
            return;
        }
        var response = JSON.parse(xhr.response);
        if (xhr.status == 200) {
            document.getElementById('card_number').value = "";
            document.getElementById('cc-exp').value = "";
            document.getElementById('cc-csc').value = "";
            document.getElementById('token').value = response.token;
            document.forms[0].submit();
        }
        else {
            alert(response.message);
        }
    });
    xhr.send(JSON.stringify(data));
}
```

[Important Points]

•This sample code which illustrates how to interact with token server is just for reference. We cannot guarantee whether it will work within your specific environment. Please feel free to customize the code to make it work within your environment and perform enough testing.

•This sample code is included in the archive of the MDK sample program which is being separately provided. Please download and refer the sample program for undefined variables etc from above sample program.